НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри

_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

“ ” 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки ***6.050103 «Програмна інженерія»***

на тему *Сервіс авторизації за допомогою стандарту OAuth2*

Виконав: студент IV курсу, групи *ІП-51 Патерило Владислав Володимирович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *Доцент к.т.н., доцент Фіногенов О.Д.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Консультант з графічної документації *Старший викладач Головченко М. М.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Рецензент:

доц., к.т.н., доц. Мелкумян К. Ю.
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Патерило Владиславу Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проекту « *Сервіс авторизації за допомогою стандарту OAuth2* »
керівник проекту *Доцент к.т.н., доцент Фіногенов Олексій Дмитрович*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с
2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Аналіз якості та тестування програмного забезпечення
4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Схема БД

2) Схема структурна класів програмного забезпечення

3) Креслення вигляду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*20.04.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*20.04.2019*	
3.	*Постановка та формалізація задачі*	*22.04.2019*	
4.	*Аналіз вимог до програмного забезпечення*	*25.04.2019*	
5.	*Алгоритмізація задачі*	*27.04.2019*	
6.	*Моделювання програмного забезпечення*	*28.04.2019*	
7.	*Обґрунтування використовуваних технічних засобів*	*30.04.2019*	
8.	*Розробка архітектури програмного забезпечення*	*05.05.2019*	
9.	*Розробка програмного забезпечення*	*15.05.2019*	
10.	*Налагодження програми*	*19.05.2019*	
11.	*Виконання графічних документів*	*25.05.2019*	
12.	*Оформлення пояснювальної записки*	*27.05.2019*	
13.	*Подання ДП на попередній захист*	*28.05.2019*	
14.	*Подання ДП рецензенту*	*01.06.2019*	
15.	*Подання ДП на основний захист*	*08.06.2019*	

Студент _____ Патерило В. В.
(підпис)

Керівник проекту _____ Фіногенов О. Д.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-5116.045490.01.81	Пояснювальна записка	68	
3	А4	КПІ.ІП-5116.045490.02.13	Опис програми	19	
4	А4	КПІ.ІП-5116.045490.03.91	Технічне завдання	11	
5	А4	КПІ.ІП-5116.045490.04.51	Програма та методика тестування	9	
6	А4	КПІ.ІП-5116.045490.05.34	Керівництво користувача	7	
7	А3	КПІ.ІП-5116.045490.06.99.СБД	Схема бази даних	1	
8	А3	КПІ.ІП-5116.045490.06.99.СС	Схема структурна класів програмного забезпечення	1	
9	А3	КПІ.ІП-5116.045490.06.99.КЕ	Схема структурна екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5116.045490.00.90		
Розробн.		Патерило В. В.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Фіногенов О. Д.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-51	
Н/контр.		Головченко М.М.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Сервіс авторизації за допомогою стандарту OAuth2

Київ – 2019 року

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 24 таблиці та 21 джерело – загалом 67 сторінок.

Об`єкт дослідження: Сервіс авторизації за допомогою стандарту OAuth2

Мета дипломного проекту: розробити програмне забезпечення, за допомогою якого можна авторизувати користувачів.

У першому розділі була вивчена теорія з даної області, переглянуто усі плюси і мінуси існуючих рішень. Було розроблено функціональні та нефункціональні вимоги.

У другому розділі було розроблено архітектуру програмного забезпечення. Побудовано схему БД та структурну схему класів.

У третьому розділі проведено тестування програмного забезпечення за розробленим планом тестування. Описано процес тестування.

У четвертому розділі описано розгортання та впровадження програмного забезпечення, а також наведено схему структурну розгортання.

У документах наведено: опис програми, схема структурна класів програмного забезпечення, схема БД, схема структурна екранних форм.

КЛЮЧОВІ СЛОВА: OAUTH2, АВТОРИЗАЦІЯ, КОРИСТУВАЧ, ДОДАТОК, ТОКЕН

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, 24 tables and 21 sources – total 67 pages.

The object of study: OAuth2 authorization service.

The aim of the diploma project: create software to authorize users.

In first section, the theory of subject was investigated, all pros and cons of existing projects. Were formed functional and unfunctional requirements.

In second section, the architecture of software was described and developed. A database scheme and a diagram of classes are constructed.

In third secion, resulting web application was tested according to the developed test plan. The process of testing is described.

In fourth section describes the deployment and implementation of microservice web applications. The diagram of structural deployment is provided.

Documents contain the description of the program, the diagram of the structural classes of the software, the scheme of database, the scheme of the structural screen forms.

KEYWORDS: OAUTH2, AUTHORIZATION, USER, APPLICATION, TOKEN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8

ВСТУП 9

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 10

1.1 Загальні положення 10
1.2 Змістовний опис і аналіз предметної області 16
1.3 Аналіз успішних IT-проектів 28
1.3.1 Аналіз відомих технічних рішень 28
1.4 Аналіз вимог до програмного забезпечення 36
1.4.1 Розроблення функціональних вимог 36
1.4.2 Розроблення нефункціональних вимог 48
1.4.3 Постановка комплексу завдань модулю 48
1.5 Висновки по розділу 49

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 50

2.1 Моделювання та аналіз програмного забезпечення 50
2.2 Архітектура програмного забезпечення 51
2.2.1 API 52
2.2.2 Адміністративна панель 56
2.3 Конструювання програмного забезпечення 57
2.3.1 Основна мова програмування 57
2.3.2 Фреймворк розробки 58
2.4 Аналіз безпеки даних 60
2.5 Висновки по розділу 61

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 62

3.1 Аналіз якості ПЗ 62
3.2 Опис процесів тестування 62
3.3 Опис контрольного прикладу 62

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 63

4.1 Розгортання програмного забезпечення 63
4.2 Робота з програмним забезпеченням 63

ВИСНОВКИ 64

ПЕРЕЛІК ПОСИЛАНЬ 65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Сервер авторизації – запитує власників ресурсів дозволи клієнтським програмам отримувати доступ до їх даних. Він також управляє і видає токени, необхідні для всіх процесів, що підтримуються стандартом OAuth2. Зазвичай одна і та ж програма, що віддає ресурси через API, захищене OAuth2, також є сервером авторизації.

Сервер ресурсів – додаток, що надає доступ до ресурсів через захищений API, який відповідає специфікації OAuth2.

Додаток – представляє клієнта на сервері авторизації. Зазвичай додаток створюється вручну розробниками клієнта після входу на сервер авторизації.

Клієнт – це додаток, уповноважений на доступ до ресурсів, захищених OAuth2, від імені та з дозволу власника ресурсу.

Власник ресурсів – користувач додатку, який надає ресурси стороннім додаткам через OAuth2. Власник ресурсу повинен дати дозвіл додаткам третьої сторони, щоб мати доступ до його даних.

Токен доступу – токен, необхідний для доступу до ресурсів, захищених OAuth2. Він має термін дії, яке зазвичай досить короткий.

Код авторизації – отримується за допомогою сервера авторизації як посередника між клієнтом і власником ресурсу. Він використовується для аутентифікації клієнта і передачі токена доступу.

Токен авторизації – токен, який сервер авторизації видає клієнтам, який можуть бути замінений на токен доступу. Він має дуже короткий термін дії, оскільки обмін повинен виконуватися незабаром після того, як користувачі нададуть дозвіл на авторизацію.

ВСТУП

Програмне забезпечення сьогодні використовується майже у всіх сферах людського життя – від соціальних мереж до складних систем у медицині. З кожним днем все більше і більше інформації про користувачів зберігається у цифровому вигляді. З однієї сторони, це надзвичайно зручно і може сильно спростити повсякденне життя та зробити його більш комфортним. Без деяких технологій ми вже не можемо уявити наше життя – соціальні мережі, електронна пошта, перегляд фільмів онлайн.

Проте такий стрімкий ріст приносить з собою купу негативних факторів. Один із найактуальніших із них – безпека особистих даних. Ні для кого не секрет, що сьогодні можна отримати несанкціонований доступ майже до всіх даних про людину, різниця тільки у кількості часу і ресурсів, які потрібно витратити на це. Дуже популярними стають обговорення щодо підвищення безпеки та покращення прийнятих стандартів у даній сфері. Проблема стала настільки широкою, що нещодавно уряд ЄС прийняв «Загальний регламент про захист даних», який передбачає певні правила обробки та зберігання персональної інформації.

Через це було вирішено створити інструмент, який би дозволив безпечно авторизувати користувачів. Стандартів існує дуже багато, проте було обрано саме OAuth2 через його надійність, підтримуваність спільнотою та зручність.

Завданням дипломної роботи є створення програмного забезпечення, яке дозволить уніфікувати процес авторизації – не тільки для веб-застосунків, а для будь-якого ПЗ, підвищить безпеку даних користувачів, а також дозволить зручно та швидко інтегрувати сторонні додатки із основним програмним забезпеченням. Компонентами готового ПЗ є API, адміністративна частина та графічний інтерфейс для керування додатками.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Сьогодні майже неможливо знайти програмне забезпечення, яке не зберігало та обробляло дані користувачів. Бази даних стрімко розвовиються, що дає ефективно працювати з великими обсягами інформації. Проте з приходом такого поняття, як авторизація, водночас з’явилася купа проблем, а саме проблеми з безпечною обробкою цих даних. Тому з’явилися такі поняття, як:

- ідентифікація - це заява про те, ким ви є. Залежно від ситуації, це може бути ім'я, адреса електронної пошти, номер облікового запису, і тд.
- аутентифікація - надання доказів, що ви насправді є той, ким ідентифікувалися (від слова "authentic" - істинний, справжній).
- авторизація - перевірка, що вам дозволено доступ до запитуваного ресурсу.

Розглянемо найпопулярніші типи авторизації.

Аутентифікація по паролю

Цей метод грунтується на тому, що користувач повинен надати username і password для успішної ідентифікації і аутентифікації в системі. Пара username / password задається користувачем при його реєстрації в системі, при цьому в якості username може виступати адреса електронної пошти користувача.

Стосовно до веб-додатків, існує кілька стандартних протоколів для аутентифікації по паролю, які буде розглянуто нижче.

HTTP authentication

Цей протокол, описаний в стандартах HTTP 1.0 / 1.1, існує дуже давно і до цих пір активно застосовується в корпоративному середовищі. Стосовно до веб-сайтів працює наступним чином:

Сервер, при зверненні неавторизованого клієнта до захищеного ресурсу, відсилає HTTP статус "401 Unauthorized" і додає заголовок "WWW-Authenticate" із зазначенням схеми і параметрів аутентифікації.

Браузер, при отриманні такої відповіді, автоматично показує діалог введення username і password. Користувач вводить деталі свого облікового запису.

У всіх наступних запитах до цього веб-сайту браузер автоматично додає HTTP заголовок "Authorization", в якому передаються дані користувача для аутентифікації сервером.

Сервер аутентифікує користувача за даними з цього заголовка. Рішення про надання доступу (авторизація) проводиться окремо на підставі ролі користувача, ACL або інших даних облікового запису.

Весь процес стандартизований і добре підтримується всіма браузерами і веб-серверами. Існує кілька схем аутентифікації, що відрізняються за рівнем безпеки:

Basic - найбільш проста схема, при якій username і password користувача передаються в заголовку Authorization в незашифрованому вигляді (base64-encoded). Однак при використанні HTTPS (HTTP over SSL) протоколу, є відносно безпечною. Опис на рисунку 1.1.

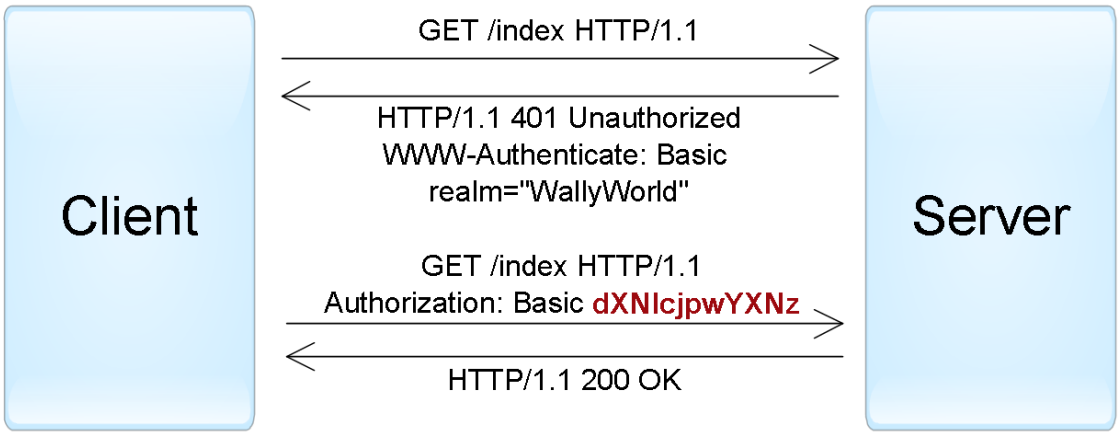


Рисунок 1.1 - Приклад HTTP аутентифікації з використанням Basic схеми.

Digest - challenge-response-схема, при якій сервер посилає унікальне значення nonce, а браузер передає MD5 хеш пароля користувача, обчислений з використанням зазначеного nonce. Більш безпечна альтернативв Basic схеми при незахищених з'єднаннях, але схильна до man-in-the-middle attacks (з заміною схеми на basic). Крім того, використання цієї схеми не дозволяє застосувати сучасні хеш-функції для зберігання паролів користувачів на сервері.

NTLM (відома як Windows authentication) - також заснована на challenge-response підході, при якому пароль не передається в чистому вигляді. Ця схема не є стандартом HTTP, але підтримується більшістю браузерів і веб-серверів. Переважно використовується для аутентифікації користувачів Windows Active Directory в веб-додатках. Вразлива до pass-the-hash-атакам.

Negotiate - ще одна схема з сімейства Windows authentication, яка дозволяє клієнтові вибрати між NTLM і Kerberos аутентифікації. Kerberos - більш безпечний протокол, заснований на принципі Single Sign-On. Однак він може функціонувати, тільки якщо і клієнт, і сервер знаходяться в зоні intranet і є частиною домену Windows.

Варто відзначити, що при використанні HTTP-аутентифікації у користувача немає стандартної можливості вийти з веб-додатки, крім як закрити всі вікна браузера.

Forms authentication

Для цього протоколу немає певного стандарту, тому всі його реалізації специфічні для конкретних систем.

Працює це за наступним принципом: в веб-додаток включається HTML-форма, в яку користувач повинен ввести свої username / password і відправити їх на сервер через HTTP POST для аутентифікації. У разі успіху веб-додаток створює session token, який зазвичай поміщається в browser cookies. При наступних веб-запитах session token автоматично передається на сервер і

дозволяє додатком отримати інформацію про поточного користувача для авторизації запиту. Опис на рисунку 1.2.

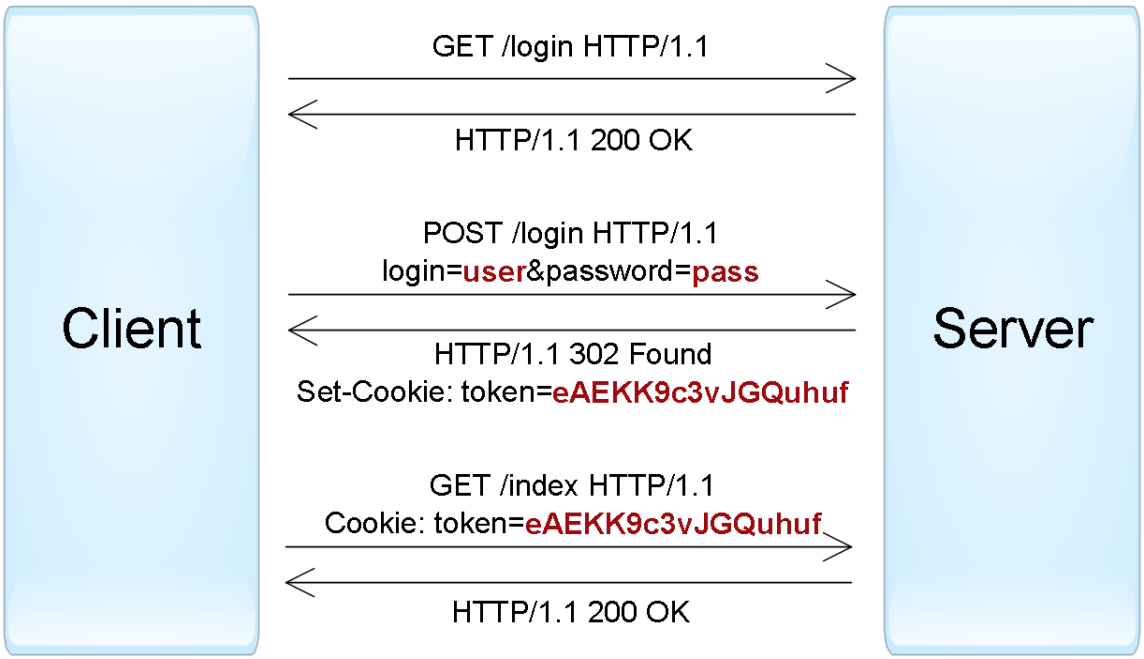


Рисунок 1.2 - Приклад forms authentication.

Додаток може створити session token двома способами:

– як ідентифікатор аутентифікованої сесії користувача, яка зберігається в пам'яті сервера або в базі даних. Сесія повинна містити всю необхідну інформацію про користувача для можливості авторизації його запитів.
– як зашифрований і / або підписаний об'єкт, що містить дані про користувача, а також вказано термін. Цей підхід дозволяє реалізувати stateless-архітектуру сервера, однак вимагає механізму поновлення сесійного токена після закінчення терміну дії.

Необхідно розуміти, що перехоплення session token часто дає аналогічний рівень доступу, що і знання username / password. Тому всі комунікації між клієнтом і сервером у разі forms authentication повинні проводитися тільки по захищеному з'єднанню HTTPS.

Поширені уразливості і помилки реалізації

Аутентифікація по паролю вважається не дуже надійним способом, так як паролі часто можна підібрати, а користувачі схильні використовувати прості і однакові паролі в різних системах, або записувати їх на клаптиках паперу. Якщо зловмисник зміг з'ясувати пароль, то користувач часто про це не дізнається. Крім того, розробники додатків можуть допустити ряд концептуальних помилок, що спрощують злом облікових записів.

Нижче представлений список найпоширеніших вразливостей в разі використання аутентифікації по паролю:

- веб-додаток дозволяє користувачам створювати прості паролі.
- веб-додаток не захищене від можливості перебору паролів (brute-force attacks).
- веб-додаток сам генерує і поширює паролі користувачам, однак не вимагає зміни пароля після першого входу (тобто поточний пароль десь записаний).
- веб-додаток допускає передачу паролів по незахищеному HTTP-з'єднання або в рядку URL.
- веб-додаток не використовує безпечні хеш-функції для зберігання паролів користувачів.
- веб-додаток не надає користувачам можливість зміни пароля або не нотифікує користувачів про зміну їх паролів.
- веб-додаток використовує вразливу функцію відновлення пароля, яку можна використовувати для отримання несанкціонованого доступу до інших облікових записів.
- веб-додаток не вимагає повторної аутентифікації користувача для важливих дій: зміна пароля, зміни адреси доставки товарів і т.д.
- веб-додаток створює session tokens таким чином, що вони можуть бути підібрані або передбачені для інших користувачів.

- веб-додаток допускає передачу session tokens по незахищеному HTTP-з'єднання, або в рядку URL.
- веб-додаток вразливе для session fixation-атак (тобто не замінює session token при переході анонімної сесії користувача в аутентифіковану).
- веб-додаток не встановлює прапори HttpOnly і Secure для browser cookies, що містять session tokens.
- веб-додаток не знищує сесії користувача після короткого періоду неактивності або не надає функцію виходу з аутентифікованої сесії.

Аутентифікація по токену

Такий спосіб аутентифікації найчастіше застосовується при побудові розподілених систем Single Sign-On (SSO), де один додаток (service provider або relying party) делегує функцію аутентифікації користувачів іншому додатку (identity provider або authentication service). Типовий приклад цього способу - вхід в додаток через обліковий запис в соціальних мережах. Тут соціальні мережі є сервісами аутентифікації, а додаток довіряє функцію аутентифікації користувачів соціальних мереж.

Реалізація цього способу полягає в тому, що identity provider (IP) надає достовірні відомості про користувача в вигляді токена, а service provider (SP) додаток використовує цей токен для ідентифікації, аутентифікації і авторизації користувача.

На загальному рівні, весь процес виглядає наступним чином:

- клієнт аутентифікующого в identity provider одним із способів, специфічним для нього (пароль, ключ доступу, сертифікат, Kerberos, …)
- клієнт просить identity provider надати йому токен для конкретного SP-додатку. Identity provider генерує токен і відправляє його клієнту.
- клієнт аутентифікується в SP-додатку за допомогою цього токена.

1.2 Змістовний опис і аналіз предметної області

OAuth 2 являє собою фреймворк для авторизації, що дозволяє додаткам здійснювати обмежений доступ до призначених для користувача аккаунтів на HTTP сервісах, наприклад, на Facebook, GitHub і DigitalOcean. Він працює за принципом делегування аутентифікації користувача сервісу, на якому знаходиться аккаунт користувача, дозволяючи сторонньому додатку отримувати доступ до аккаунту користувача. OAuth 2 працює в інтернеті, на десктопних і мобільних додатках.

Ролі OAuth

OAuth визначає чотири ролі:

- власник ресурсу
- клієнт
- сервер ресурсів
- авторизаційний сервер

Власник ресурсу: Користувач

Власником ресурсу є користувач, який авторизує додаток для доступу до свого облікового запису. Доступ до програми до призначеного для користувача аккаунту обмежений "областю видимості" (scope) наданих прав авторизації (наприклад, доступ на читання або запис).

Сервер ресурсів / авторизації: API

Сервер ресурсів безпосередньо зберігає захищені дані акаунтів користувачів, а авторизаційний сервер перевіряє справжність інформації, наданої користувачем, а потім створює авторизовані токени для програми, за допомогою яких додаток буде здійснювати доступ до призначених для користувача даних.

З точки зору розробника додатка API сервісу одночасно виконує і роль сервера ресурсів і роль сервера авторизації. Далі ми будемо вважати ці дві ролі однієї, і називати її Сервіс або API.

Клієнт: Додаток

Клієнтом є додаток, яке хоче здійснити доступ до аккаунту користувача. Перед здійсненням доступу додаток повинен бути авторизовано користувачем, а авторизація повинна бути схвалена з боку API.

Абстрактний опис протоколу

Тепер, коли у нас є уявлення про ролі, які використовуються в OAuth, розглянемо діаграму їх взаємодії один з одним. Опис на рисунку 1.3.

Рисунок 1.3 - Абстрактний опис протоколу

Розглянемо опис послідовності кроків на цій діаграмі:

- додаток запитує у користувача авторизацію на доступ до сервера ресурсів.
- якщо користувач авторизує запит, додаток отримує дозвіл на авторизацію (authorization grant).
- додаток запитує авторизаційний токен у сервера авторизації (API) шляхом надання інформації про самого себе і дозвіл на авторизацію від користувача.
- якщо справжність додатку підтверджена і дозвіл на авторизацію отримано, сервер авторизації (API) створює токен доступу для програми. Процес авторизації завершений.
- додаток запитує ресурс у сервера ресурсів (API), надаючи при цьому токен доступу для аутентифікації.
- якщо токен дійсний, сервер ресурсів (API) надає запитуваний ресурс додатку.

Фактичний порядок кроків описаного процесу може відрізнятися в залежності від використовуваного типу дозволу на авторизацію, але в цілому процес буде виглядати описаним чином. Далі ми розглянемо різні типи дозволів на авторизацію.

Реєстрація додатку

Перед тим, як почати використовувати OAuth в нашому додатку, необхідно зареєструвати своЇ додатки в сервісі. Це робиться шляхом реєстрації в розділі "developer" або "API" сайту сервісу, де необхідно надати наступну інформацію (можливо, включаючи деякі деталі про ваш додаток):

- назва додатку
- сайт додатку
- Redirect URL або callback URL

Redirect URL - це URL, на який сервіс буде перенаправляти користувача після авторизації (або відмови в авторизації) додатку.

Ідентифікатор клієнта і секрет клієнта

Після реєстрації додату сервіс створить облікові дані клієнта - ідентифікатор клієнта (client ID) і секрет клієнта (client secret). Ідентифікатор клієнта представляж собою публічно доступний рядок, який використовується API сервісом для ідентифікації додатка, а також використовується для створення авторизаційних URL для користувачів. Секрет клієнта використовується для аутентифікації справжності додатку для API сервісу, коли додаток запитує доступ до аккаунту користувача. Секрет клієнта повинен бути відомий тільки додатку і API.

Дозвіл на авторизацію

В абстрактному описі протоколу вище перші чотири кроки стосуються питань створення дозволу на авторизацію і токена доступу. Тип дозволу на авторизацію залежить від використовуваного додатком методу запиту авторизації, а також від того, які типи дозволу підтримуються з боку API. OAuth 2 визначає чотири різних типи, кожен з яких корисний в певних ситуаціях:

- код авторизації (Authorization Code): використовується з серверними додатками (server-side applications).
- неявний (Implicit): використовується мобільними або веб-додатками (додатки, що працюють на пристрої користувача).
- облікові дані власника ресурсу (Resource Owner Password Credentials): використовуються довіреними додатками, наприклад додатками, які є частиною самого сервісу.
- облікові дані клієнта (Client Credentials): використовуються при доступі додатку до API.

Далі ми розглянемо ці типи дозволу на авторизацію, приклади їх використання.

Тип дозволу на авторизацію: Код авторизації

Код авторизації є одним з найбільш поширених типів дозволу на авторизацію, оскільки він добре підходить для серверних додатків, де вихідний код програми і секрет клієнта не доступні стороннім. Процес в даному випадку будується на перенаправленні (redirection), що означає, що програма має бути в стані взаємодіяти з призначеним для користувача агентом (user-agent), наприклад, веб-браузером, і отримувати коди авторизації API, перенаправляти через призначений для користувача агент.

Процес описано на рисунку 1.4.

Рисунок 1.4 - Опис протоколу з логіном та паролем

Крок 1: Посилання з кодом авторизації

Спочатку користувачеві надається посилання наступного виду:

https://example.com/v1/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read

Розглянемо компоненти посилання:

– https:// example.com /v1/oauth/authorize: вхідні точка API авторизації (API authorization endpoint).
– client_id = CLIENT_ID: ідентифікатор клієнта додатка (за допомогою цього ідентифікатора API розуміє, яке додаток запитує доступ).
– redirect_uri = CALLBACK_URL: URL, на який сервіс перенаправить користувача агент (браузер) після видачі авторизационного коду.
– response_type = code: вказує на те, що додаток запитує доступ за допомогою коду авторизації.
– scope = read: задає рівень доступу додатку (в даному випадку - доступ на читання).

Крок 2: Користувач авторизує додаток

Коли користувач натискає на посилання, він повинен спершу здійснити вхід в систему для підтвердження своєї особи (якщо він, звичайно, ще не залогінений). Після цього сервіс запропонує користувачеві авторизувати або відмовити в авторизації додатком для доступу до аккаунту користувача. Приклад такого діалогу представлений на рисунку 1.5.

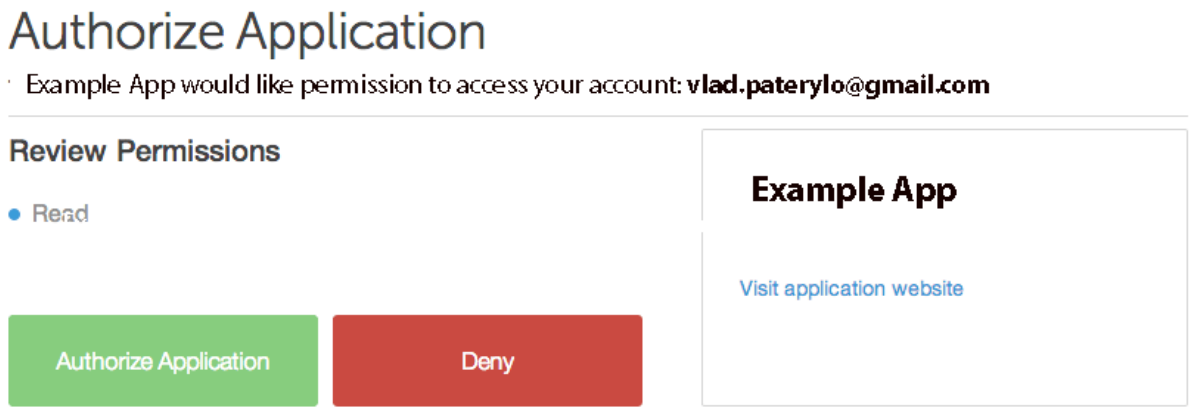


Рисунок 1.5 - користувач авторизує додаток

Крок 3: Додаток отримує код авторизації

Якщо користувач вибирає "Авторизувати додаток", сервіс перенаправляє користувача агент (браузер) по URL перенаправлення (redirect URL), який був заданий на етапі реєстрації клієнта (разом з кодом авторизації). Посилання буде виглядати схожим чином (в даному прикладі додаток називається "redirect.com"):

https://redirect.com/callback?code=AUTHORIZATION_CODE

Крок 4: Додаток запитує токен доступу

Додаток запитує токен доступу у API шляхом відправки авторизаційного коду і аутентифікаційної інформації (включаючи секрет клієнта) сервісу. Нижче представлений приклад POST-запиту для отримання токена Example.com:

https://example.com/v1/oauth/token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHORIZATION_CODE&redirect_uri=CALLBACK_URL

Крок 5: Додаток отримує токен доступу

Якщо авторизація пройшла успішно, API повертає токен доступу (а також, опціонально, токен для поновлення токена доступу - refresh token). Вся відповідь сервера може виглядати наступним чином:

{ "Access_token": "ACCESS_TOKEN", "token_type": "bearer", "expires_in": 2592000, "refresh_token": "REFRESH_TOKEN", "scope": "read", "uid": 100101, "info": { "name": “Vlad Paterylo", "email": "vlad.patertylo@gmail.com"}}

Тепер додаток авторизовано. Воно може використовувати токен для доступу до призначеного для користувача аккаунту через API сервісу з заданими обмеженнями доступу до тих пір, поки не закінчиться термін дії токена або токен не буде відкликано. Якщо був створений токен для

поновлення токена доступу, він може бути використаний для отримання нових токенів.

Тип дозволу на авторизацію: Неявний

Неявний тип дозволу на авторизацію використовується мобільними і веб-додатками (програмами, які працюють в веб-браузері), де конфіденційність секрету клієнта не може бути гарантована. Неявний тип дозволу також заснований на перенаправленні користувацького агента, при цьому токен доступу передається призначеному для користувача агенту для подальшої передачі з додатком. Це, в свою чергу, робить токен доступним користувачеві і іншим додаткам на пристрої користувача. Також при цьому типі дозволу на авторизацію не провадиться аутентифікація справжності додатки, а сам процес покладається на URL перенаправлення (зареєстрований раніше в сервісі).

Неявний тип дозволу на авторизацію не підтримує токени поновлення токена доступу (refresh tokens).

Процес виглядає наступним чином: додаток просить користувача авторизувати себе, потім сервер авторизації передає токен доступу до призначеного для користувача агенту, який передає токен з додатком. Опис наведено на рисунку 1.6.

Рисунок 1.6 - Опис неявного дозволу на авторизацію

Крок 1: Посилання для неявної авторизації

При неявному типу дозволу на авторизацію користувачеві надається посилання, яке запитує токен у API. Це посилання виглядає майже так само, як посилання для попереднього способу (з кодом авторизації), за винятком того, що запитується токен замість коду (зверніть увагу на response type "token"):

https://example.com/v1/oauth/authorize?response_type=token&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read

Крок 2: Користувач авторизує додаток

Коли користувач натискає на посилання, він повинен спершу здійснити вхід в систему для підтвердження своєї особи (якщо він, звичайно, ще не

авторизований). Після цього сервіс запропонує користувачеві авторизувати або відмовити в авторизації додатку для доступу до аккаунту користувача. Приклад такого діалогу представлений нижче:

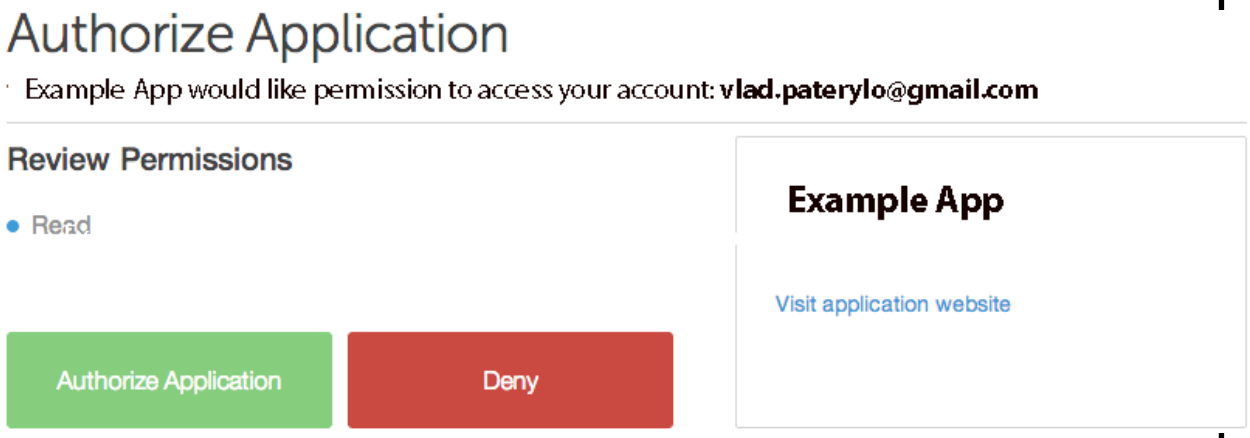


Рисунок 1.7 - користувач авторизує додаток

Крок 3: Призначений для користувача агент отримує токен доступу з URI перенаправлення

Якщо користувач вибирає "Авторизувати додаток", сервіс перенаправляє користувацький агент по URI перенаправлення додатку і включає в URI фрагмент, що містить токен доступу. Це виглядає приблизно ось так:

https://maapp.com/callback#token=ACCESS_TOKEN

Крок 4: Користувацький агент переходить по URI перенаправлення, зберігаючи при цьому токен доступу.

Крок 5: Додаток виконує скрипт вилучення токена доступу.

Додаток повертає веб-сторінку, яка містить скрипт для вилучення токену доступу з повного URI перенаправлення, збереженого призначеним для користувача агентом.

Крок 6: Токен доступу передається додатку

Призначений для користувача агент запускає скрипт вилучення токена доступу, а потім передає вилучений токен додатку.

Тепер додаток авторизовано. Воно може використовувати токен для доступу до призначеного для користувача аккаунту через API сервісу з заданими обмеженнями доступу до тих пір, поки не закінчиться термін дії токена або токен не буде відкликано.

Тип дозволу на авторизацію: облікові дані власника ресурсу

При цьому типу дозволу на авторизацію користувач надає додатку безпосередньо свої авторизовані дані в сервісі (ім'я користувача і пароль). Додаток, в свою чергу, використовує отримані облікові дані користувача для отримання токена доступу від сервісу. Цей тип дозволу на авторизацію повинен використовуватися тільки в тому випадку, коли інші варіанти не доступні. Крім того, цей тип дозволу варто використовувати тільки в разі, коли додаток користується довірою користувача (наприклад, є частиною самого сервісу, або операційної системи користувача).

Процес з обліковими даними власника ресурсу:

Після того, як користувач передасть свої облікові дані додатку, додаток запросить токен доступу у авторизационного сервера. Приклад POST-запиту може виглядати наступним чином:

https://oauth.example.com/token?grant_type=password&username=USERNAME&password=PASSWORD&client_id=CLIENT_ID

Якщо облікові дані коректні, сервер авторизації поверне токен доступу з додатком. Тепер додаток авторизовано.

Тип дозволу на авторизацію: Облікові дані клієнта

Тип дозволу на авторизацію з використанням облікових даних клієнта дозволяє додатку здійснювати доступ до свого власного аккаунту сервісу. Це може бути корисно, наприклад, коли додаток хоче оновити власну реєстраційну інформацію на сервісі або URI перенаправлення, або ж здійснити доступ до іншої інформації, що зберігається в акаунті додатку на сервісі, через API сервісу.

Процес з обліковими даними клієнта

Додаток запитує токен доступу шляхом відправки своїх облікових даних, ідентифікатора клієнта і секрету клієнта авторизационному серверу. Приклад POST-запиту може виглядати наступним чином:

https://oauth.example.com/token?grant_type=client_credentials&client_id=CLIENT_ID&client_secret=CLIENT_SECRET

Якщо облікові дані клієнта коректні, авторизаційний сервер поверне токен доступу для програми. Тепер додаток авторизовано для використання власного облікового запису!

Приклад використання токена доступу

Після того, як додаток отримає токен доступу, воно може використовувати цей токен для доступу до призначеного для користувача аккаунту через API сервісу з заданими обмеженнями доступу до тих пір, поки не закінчиться термін дії токена або токен не буде відкликано.

Нижче представлений приклад запиту до API з використанням curl. Зверніть увагу, що він містить токен доступу:

curl -X GET -H "Authorization: Bearer ACCESS_TOKEN" "https://api.example.com/v2/user/ "

Якщо токен доступу валідний, API обробить отриманий запит. Якщо термін дії токена доступу закінчився або токен є некоректним, API поверне помилку "invalid_request".

Оновлення токена доступу

Після закінчення терміну дії токена доступу всі запити до API з його використанням повертатимуть помилку "Invalid Token Error". Якщо при створенні токена доступу був створений і токен для поновлення токена доступу (refresh token), останній може бути використаний для отримання нового токена доступу від авторизационного сервера.

Нижче представлений приклад POST-запиту, що використовує токен для поновлення токена доступу для отримання нового токена доступу:

https://example.com/v1/oauth/token?grant_type=refresh_token&client_id=CLIENT_ID&client_secret=CLIENT_SECRET&refresh_token=REFRESH_TOKEN

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Microsoft Azure Active Directory Review

Плюси: Найкраща інтеграція з Windows Server Active Directory. Тісна інтеграція з хмарними сервісами Microsoft. Захист даних такий, що дозволяє застосовувати політику безпеки на основі BigData та машинного навчання (ML).

Мінуси: Деякі конкуренти мають кращу інтеграцію з каталогами сторонніх виробників і платформами SaaS. Додаткові можливості звітності доступні лише у рівнях преміум-ціноутворення.

Підсумок: Microsoft Azure Active Directory (AD) отримує своїх клієнтів у сфері Identity-Management-as-a-Service (IDaaS) через тісну інтеграцію з Windows Server Active Directory і Office 365. Azure AD також пропонує найнижчий початковий рівень ціноутворення для обробки багатофакторної автентифікації та пропонує розширені набори інструментів для керування ідентифікаційними даними та різноманітні хмарні програми.

Okta Identity Management Review

Плюси: Підтримка управління мобільними пристроями (MDM) і географічних зон роблять Okta достойним конкурентом на ринку. Функціональність звітів значно покращилася, зокрема географічна функціональність. Можливість керувати потоком інформації користувача між кількома провайдерами даних є одним з кращих у цій категорії.

Мінуси: Для автентифікації на локальних програмах потрібне дороге обладнання.

Підсумок: Не дивно, що Okta Identity Management настільки шанований в сфері Identity-Management-as-a-Service (IDaaS). Маючи різноманітні функції, які включають в себе політики безпеки, підтримку MDM та геолокацію, можливість інтегрувати декілька джерел ідентифікаційних даних, і всі вони упаковані в рішення, яке є відносно простим у використанні, робить Okta Identity Management одним з кращих рішень IDaaS на ринку.

EmpowerID Review

Плюси: Локальна установка забезпечує більшу гнучкість. Покращена безпека для локальної архітектури. Комплексна функціональність звітності.

Мінуси: витрати на керування роботою та встановленням значно вищі порівняно з хмарними рішеннями. Веб-сайт для мобільних пристроїв не є відповідною заміною для мобільних програм для всіх організацій.

Підсумок: EmpowerID пропонує комплексне рішення IDaaS (Identity-Management-as-a-Service) як для керування ідентифікаційними даними в Інтернеті, так і в межах існуючого корпоративного каталогу, але зі значним збільшенням як початкової складності, так і поточних вимог до обслуговування.

OneLogin Review

Плюси: Інтерфейс допомагає оптимізувати управління користувачами та ролями. Проксі-агенти пропонують легку підтримку локальних додатків. Налаштування сповіщень електронною поштою просте.

Мінуси: групи AD не синхронізовані. Розширені функції доступні тільки за високу плату.

Підсумок: OneLogin має хороший набір функцій, включаючи політику аутентифікації на основі ризиків, інтеграцію з HR-програмами та платформи моніторингу подій.

Optimal IdM Review

Плюси: Найвищий рівень обслуговування вимагає менше технічних знань від клієнтів, ніж інші системи. Конфігурація приватної хмари забезпечує безпеку, продуктивність і надійність. Virtual Identity Server пропонує спрощений метод обслуговування корпоративних даних з різних джерел. Брандмауер LDAP дозволяє розділяти програми та сховище даних.

Мінуси: ціноутворення звужує клієнтську базу до великих підприємств. Нульова інтеграція з SaaS. Обмежені можливості користувачів налаштовувати інтерфейс SSO.

Підсумок: Optimal IdM має всі необхідні функціх, необхідні в Identity-Management-as-a-Service (IDaaS), але за високу ціну. Оскільки щомісячні витрати легко можуть бути у діапазоні $ 25 000 - $ 30 000, більшість підприємств роблять вибір на кориусть одного із конкурентів, таких як Microsoft Azure Active Directory та Okta Identity Management плюс один або два штатних співробітники.

Bitium Review

Плюси: Можливість використовувати Google SSO. Функції самообслуговування, такі як скидання пароля мобільного телефону, можуть заощадити час і гроші. Закладки з переходом на SaaS додатки полегшують життя користувачам.

Мінуси: Немає підтримки IDM для клієнта (consumer). Обмежена кількість існуючих корпоративних програм. Підтримка кількох джерел даних відстає від лідерів галузі.

Підсумок: Bitium пропонує безліч приємних можливостей для користувачів, включаючи скидання пароля мобільного телефону та закладки для певних місць у додатках SaaS інших виробників. На жаль,

найважливіші функції для адміністраторів не на найкращому рівні: SSO через Google має багато користі, але відсутність підтримки даних споживачів і обмежені набори інструментів для декількох каталогів є потенційною проблемою при виборі даного сервісу серед конкурентів.

Centrify Identity Service Review

Плюси**:** Повнофункціональні можливості звітності, включаючи дашборд. Локальна програма проста у використанні та використовує той же програмний агент, що й AD. Швидка інтеграція з даними користувачів з соціальних мереж. Аутентифікація може використовувати машинне навчання за додаткову плату.

Мінуси: Нездатність посилатися на користувачів AD та групи. Сценарії вимагають набір навичок рівня розробника.

Підсумок: Centrify пропонує функції, які просто не пропонуються конкурентами, а також предоставляю широкий спектр функцій, такі як звітність, підтримка даних споживачів і легкий доступ до локальних додатків.

Ping Identity PingOne Review

Плюси: Налаштування проводиться відносно легко. Каталог програм є комплексним для цілей SSO. Здійснення зв’язування програм із групами займає не більше хвилини.

Мінуси: Підтримка забезпечення SaaS не поширюється на Microsoft Office 365. Обмежені можливості звітності.

Підсумок: Інтеграція із SaaS є найбільш слабким місцем, хоча і не повністю відсутнє.

LastPass Enterprise Review

Плюси: Величезна кількість політик безпеки. Спільні папки надають користувачам можливість самостійно керувати деякими загальними обліковими даними. Splunk інтеграція та повідомлення електронною поштою роблять легким моніторинг.

Мінуси: Інтеграція AD забезпечує мінімум функціональності. Забезпечення SAML підтримує лише декілька програм і відсутні деякі ключові параметри. Можливості звітності обмежені.

Підсумок: LastPass Enterprise зробив значні поліпшення за останні два роки, але все ще відстає від конкурентів у ключових категоріях. Тим не менш, функції, такі як спільні папк, роблять LastPass Enterprise правдоподібним для малих підприємств.

Identacor Review

Плюси: Одноразовий вхід дозволяє зберігати локальний трафік аутентифікації, коли запит ініціюється в корпоративній мережі. Ціна початкового рівня справедлива, особливо з використанням SaaS.

Мінуси: Керування групами вручну - це взагалі неправильно для керування хмарними застосунками. Немає підтримки каталогів LDAP. Багатофакторні параметри дуже обмежені.

Підсумок: В Identacor відсутні багато можливостей, щоб вважати його основним претендентом у сфері Identity-Management-as-a-Service (IDaaS). Динамічні групи є найяскравішим упущенням, але відсутність опцій для багатофакторної аутентифікації і навіть підтримка сторонніх каталогів LDAP є іншими помітними моментами, які потребують роботи.

VMware Workspace One Review

Плюси: Інтеграція ключів з AirWatch дозволяє забезпечити синхронізовані пристрою. Можливість забезпечення аутентифікації віртуальним додаткам або настільним комп'ютерам за допомогою інтеграції Horizon. Політики дозволяють використовувати різні комбінації методів аутентифікації, забезпечуючи підтримку багатофакторної або запасної аутентифікації.

Мінуси: Інтеграція з Active Directory або LDAP вимагає конфігурації декількох шарів. Інструменти можуть не відповідати мінімальним вимогам для деяких підприємств.

Висновок: Ключові інтеграції з AirWatch та Horizon роблять даний продукт дуже привабливим для компаній, які вже інвестували в екосистему VMware. Недоліки в таких областях, як звітність або спрощене налаштування, можуть мати протилежний вплив на потенційних клієнтів, які ще не мають інвестицій у VMware.

PortalGuard Review

Плюси: Ціни дуже конкурентоспроможні для корпоративних клієнтів з великими базами користувачів. Локальна установка пропонує додатковий рівень контролю. Аутентифікація на основі ризику дозволяє застосовувати складні правила політики аутентифікації.

Мінуси: Повністю відсутня будь-яка можливість інтеграції SaaS. Відсутність підтримки даних споживачів.

Підсумок: відсутність інтеграції з SaaS. Ціноутворення для великих підприємств є конкурентоспроможним і може зробити

PortalGuard вартою уваги на підприємств, яким не обов'язково потрібні SaaS.

Порівняння готових рішень наведено у таблиці 1.1.

Продукт	OneLogin	Centrify	Microsoft	Okta	VMware	EmpowerID	Optimal	Bitium	LastPass Enterprise	Ping Identity PingOne
Комплексна бібліотека звітів	+	+	+	+	-	+	+	-	-	-
Під'єднання каталогу	+	+	+	+	+	+	+	+	-	+
Декілька правил для SSO	+	+	+	+	+	+	+	-	+	-
Синхронізація пароля	+	-	+	+	-	-	-	-	+	-
Настроювана користувачем SSO панель	+	-	+	+	-	-	-	-	-	-
Користувач може сам редагувати	+	-	+	+	+	+	+	-	-	-
Інтеграція з SaaS	+	+	+	+	+	+	+	+	-	+
Сторонні мультифакторні провайдери	+	-	-	+	+	+	+	-	+	-
Мобільна SSO	+	+	+	+	+	+	+	+	+	+
SAML аутентифікація	+	+	+	+	+	+	+	+	+	+
REST API	+	+	+	+	+	+	+	+	+	+
Аутентифікація до офлайн-додатків	+	+	+	+	+	+	+	+	-	+
Інтеграція зі сторонніми MDM	+	-	+	+	+	-	-	+	-	-
Інтеграція декількох каталогів	+	+	+	+	+	+	+	+	-	+

Таблиця 1.1 - таблиця порівняння всіх готових рішень

1.4 Аналіз вимог до програмного забезпечення

Розроблене програмне забезпечення повинно мати наступний функціонал:

- отримання Authorization Code додатком
- отримання токена додатком
- отримання токена за допомогою логіна і пароля користувача
- оновлення токена доступу
- видалення токена доступу
- обмеження доступу до певних API відповідно до наданих додатку дозволів (scopes)
- перевірка валідності токена
- отримання списку токенів
- видалення токену через адміністративну частину додатку
- редагування даних додатку через графічний інтерфейс

1.4.1 Розроблення функціональних вимог

При розробці ПЗ було передбачено наступні варіанти використання:

Таблиця 1.2 – Варіант використання UC001

Назва	Отримання Authorization Code додатком
Опис	Сторонні додатки повинні отримати Authorization Code аби після цього отримати токен доступу
Учасники	Сторонній додаток, Авторизаційний сервер, Користувач
Передумови	Додаток зареєстровано і він має свої Client ID та Client Secret
Постумови	Додаток отримав Authorization Code
Основний сценарій	Додаток запитує у користувача дозвіл до певної інформації профілю, користувач дає дозвіл, далі додаток виконує API запит, де вказує необхідні параметри

Таблиця 1.3 – Варіант використання UC002

Назва	Отримання токена додатком
Опис	Сторонні додатки отримують токен для подальшого доступу до інформації профіля користувача
Учасники	Сторонній додаток, Авторизаційний сервер, Користувач
Передумови	Додатком отримано Authorization Code

продовження таблиці 1.3

Постумови	Додаток отримав токен доступу
Основний сценарій	Додаток виконує API запит на авторизаційний сервер, вказуючи Authorization Code, отриманий перед цим. Якщо всі параметри правильні, додаток отримує відвовідь від сервера, в якому міститься токен доступу

Таблиця 1.4 – Варіант використання UC003

Назва	Отримання токена за допомогою логіна і пароля користувача
Опис	Довірені додатки, як наприклад внутрішній сервіс, можуть авторизовувати користувачів у своїх додатках за допомгою логіна та пароля користувача
Учасники	Додаток, Авторизаційний сервер, Користувач
Передумови	Додаток хоче авторизувати користувача
Постумови	Додаток отримав токен доступу
Основний сценарій	Користувач вводить свої логін та пароль, далі додаток виконує API запит на авторизаційний сервер. Якщо дані правильні, то у відповіді буде міститись токен доступу.

Таблиця 1.5– Варіант використання UC004

Назва	Оновлення токена доступу
Опис	Токен доступу має термін дії, який зазвичай доволі короткий, наприклад 1 година. Через це його потрібно оновлювати
Учасники	Сторонній додаток, Авторизаційний сервер
Передумови	Токен доступу недійсний і у додатка є відповідний токен оновлення
Постумови	Токен доступу та токен оновлення дійсні та були замінені на нові
Основний сценарій	Додаток повинен виконати запит до авторизаційного серверу, вказавши токен доступу, термін дії якого закінчився, а також відповідний токен оновлення

Таблиця 1.6 – Варіант використання UC005

Назва	Видалення токена доступу
Опис	Авторизаційний сервер дає можливість видалити токен доступу
Учасники	Сторонній додаток, Авторизаційний сервер
Передумови	Додаток має дійсний токен, який хоче видалити
Постумови	Токен доступу видалено

продовження таблиці 1.6

Основний сценарій	Додаток виконує API запит на авторизаційний сервер, вказуючи токен доступу, який потрібно видалити.

Таблиця 1.7 – Варіант використання UC006

Назва	Обмеження доступу до певних API відповідно до наданих додатку дозволів (scopes)
Опис	Доступ до ресурсів може бути обмежений різними способами. Наприклад дозвіл тільки на читання або запис. Розробник може створювати свої дозволи в коді авторизаційного сервера.
Учасники	Авторизаційний сервер, Розробник
Передумови	Розробник хоче обмежити доступ до ресурсу «Номер телефону користувача» - тільки читання
Постумови	Доступ до ресурсу «Номер телефону користувача» обмежено
Основний сценарій	Розробник додає потрібні дозволи на endpoint API

Таблиця 1.8 – Варіант використання UC007

Назва	Перевірка валідності токена
Опис	Додаток може перевіряти токени на їх валідність
Учасники	Сторонній додаток, Авторизаційний сервер

продовження таблиці 1.8

Передумови	Додатком має певний токен доступу
Постумови	Додаток отримав інформацію щодо токену
Основний сценарій	Додаток виконує API запит на авторизаційний сервер, вказуючи токен доступу. У відповіді сервера є інформація про те, чи дійсний токен, а також його термін дії та ще деякі параметри.

Таблиця 1.9 – Варіант використання UC008

Назва	Отримання списку токенів
Опис	Адміністратор авторизаційного сервера може переглядати список наявних токенів
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера
Передумови	Адміністратор має доступ до авторизаційного сервера
Постумови	Адміністратор отримав список наявних токенів
Основний сценарій	Адміністратор повинен перейти у адміністративну панель, перейти за посиланням за написом «Токени»

Таблиця 1.10 – Варіант використання UC009

Назва	Видалення токену через адміністративну частину додатку

продовження таблиці 1.10

Опис	Адміністратор авторизаційного сервера може видаляти токени
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера
Передумови	Адміністратор авторизаційного сервера хоче видалити токен
Постумови	Токен видалено
Основний сценарій	Адміністратор повинен перейти до адміністративної частини сайту, потім на сторінку зі списком токенів, обрати на певний токен і на сторінці токена натиснути на «Видалити»

Таблиця 1.11 – Варіант використання UC010

Назва	Отримання списку додатків
Опис	Адміністратор авторизаційного сервера може отримувати доступ до усіх зареєстрованих додатків
Учасники	Адміністратор авторизаційного сервера, Авторизаційний сервер
Передумови	Адміністратор авторизаційного сервера хоче переглянути список зареєстрованих додатків
Постумови	Адміністратор авторизаційного сервера отримав список зареєстрованих додатків

продовження таблиці 1.11

Основний сценарій	Адміністратор авторизаційного сервера повинен перейти до адміністративної панелі сайту, далі перейти на сторінку зі списком додатків.

Таблиця 1.12 – Варіант використання UC011

Назва	Створення нового додатку
Опис	Авторизаційний сервер предоставляє графічний інтерфейс для створення нових додатків
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера
Передумови	Потрібно створити новий додаток
Постумови	Додаток створено
Основний сценарій	Адміністратор авторизаційного сервера вводить необхідні дані у форму. Якщо все введено правильно, створюється додаток і відповідно його client id та client secret

Таблиця 1.13 – Варіант використання UC012

Назва	Редагування даних додатку
Опис	Авторизаційний сервер предоставляє графічний інтерфейс для редагування додатків
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера

продовження таблиці 1.13

Передумови	Необхідно змінити інформацію щодо додатку
Постумови	Інформація додатка оновлена
Основний сценарій	Адміністратор авторизаційного сервера вносить необхідні зміни даних додатку у форму. Якщо все введено правильно, інформація додатку оновлюється.

Таблиця 1.14 – Варіант використання UC013

Назва	Створення користувачів
Опис	Адміністратор авторизаційного сервера має можливість створювати користувачів
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера
Передумови	Адміністратор авторизаційного сервера має дані користувача, якого потрібно створити
Постумови	Користувача створено
Основний сценарій	Адміністратор авторизаційного сервера вводить усі дані користувача в адміністративній панелі, натискає кнопку «Створити»

Таблиця 1.15 – Варіант використання UC014

Назва	Редагування даних користувачів
Опис	Адміністратор авторизаційного сервера має можливість редагувати дані користувачів
Учасники	Авторизаційний сервер, Адміністратор авторизаційного сервера
Передумови	Адміністратор авторизаційного сервера хоче змінити дані користувача
Постумови	Дані користувача оновлено
Основний сценарій	Адміністратор авторизаційного сервера переходить на сторінку зі списком усіх користувачів, далі знаходить потрібного и натискає на нього. Відкривається сторінка з усіма даними користувача, які можна змінити. Після змін потрібно натиснути на кнопку «Зберегти»

Вимоги до користувацького модулю розв’язку тестів описано наступними таблицями:

Таблиця 1.16 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Вхід до застосунку
Опис	Перед початком роботи система має запрошувати авторизацію користувача. Після авторизації користувач володіє усім функціоналом системи.

Таблиця 1.17 – Опис функціональної вимоги REQ001

Номер	REQ002
Назва	Перегляд доступних додатків
Опис	Система відібражає доступні додатки, якщо перейти на сторінку додатків

Таблиця 1.18 – Опис функціональної вимоги REQ001

Номер	REQ003
Назва	Можливість створення токену
Опис	За допомогою API можна створини токен доступу. Також це можна зробити через адміністративну панель

Таблиця 1.19 – Опис функціональної вимоги REQ001

Номер	REQ004
Назва	Можливість створення додатку
Опис	За допомогою графічного інтерфейсу керування додатками можна створити новий додаток

Таблиця 1.20 – Опис функціональної вимоги REQ001

Номер	REQ005
Назва	Можливість редагування додатку
Опис	За допомогою графічного інтерфейсу керування додатками можна редагувати існуючий додаток

Таблиця 1.21 – Опис функціональної вимоги REQ001

Номер	REQ006
Назва	Можливість видалення додатку
Опис	За допомогою графічного інтерфейсу керування додатками можна видалити існуючий додаток. Це також можна зробити через адміністративну панель сайту.

Таблиця 1.22 – Опис функціональної вимоги REQ001

Номер	REQ007
Назва	Можливість видалення токенів
Опис	За допомогою графічного можна видалити токен. Це також можна зробити через API і адміністративну панель ПЗ.

Взаємозв’язки між вимогами клієнтського застосунку і варіантами використання відображено на рисунку 1.8.

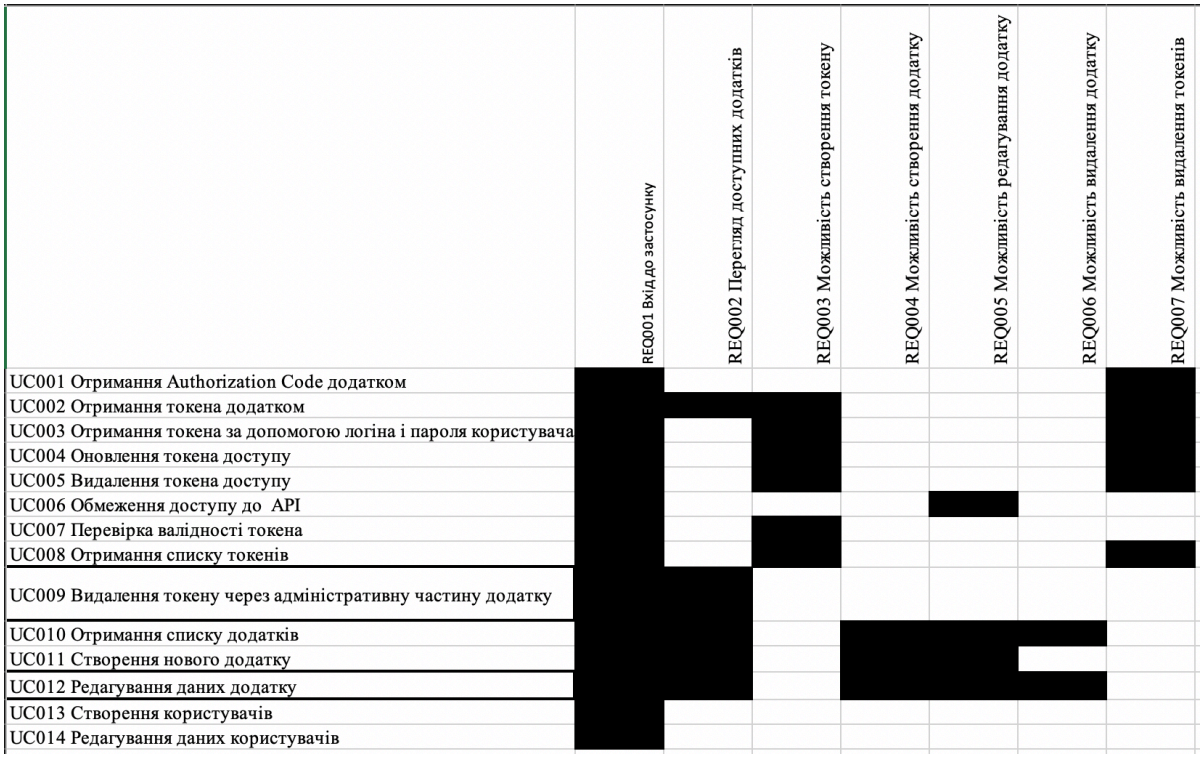


Рисунок 1.8 - Матриця залежності між вимогами застосунку і варіантами використання

1.4.2 Розроблення нефункціональних вимог

Сервіс авторизації за допомогою стандарту OAuth2 повинен відповідати наступним нефункціональним вимогам:

- локалізація інтерфейсу – повинні підтримуватись англійська та українська мови
- формат даних, які передаються на сервер – JSON
- при розробці API повинен бути використаний підхід до архітектури мережевих протоколів - REST

1.4.3 Постановка комплексу завдань модулю

Розроблене програмне рішення повинно повністю відповідати за авторизацію користувачів незалежно від платформи розробки – WEB, Mobile, Embedded, Desktop та ін.

Метою розробки є підвищення безпеки користувачів, зручний та уніфікований процес авторизації. Підвищення ефективності розробки в цілому – а саме інтегрування різних мікросервісів з основним проектом. Надалі для отримання доступу мікросервісами до даних про користувача буде використано тільки одна точка входу. Створення сервісу для авторизації користувачів, який буде складатися з API для керування користувачами, певного графічного інтерфейсу, а також адміністративної панелі. В результаті використання готового рішення будуть значно знижені ресурси на розробку ПЗ та підвищена продуктивність розробників. Головне – стандарт OAuth2 має доволі високий рівень безпеки, тож персональні дані користувачів будуть захищені від несанкціонованого доступу.

1.5 Висновки по розділу

Зберігання та обробка даних користувачів стала дуже важливою частиною будь-якого програмного забезпечення. З кожним року цій темі придається все більше і більше уваги не тільки розробників, а й спільноти. Обране технічне рішення дозволяє ефективно та безпечно контролювати авторизацію користувачів, а також відкриває нові можливості для інтеграції зі сторонніми додатками, що однозначно посприяє розвитку проекту.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для того, щоб створити ефективне та надійне програмне забезпечення, потрібно спочатку проаналізувати його архітектуру. Як було зазначено вище, ПЗ буде складатися з трьох частин: API, адміністративна панель та графічний інтерфейс для керування додатками.

Для того, щоб отримати доступ до API, користувач повинен мати токен доступу. Якщо в нього немає токена зовсім або токен оновлення недійсний, то він повинен отримати заново. Опис наведено на рисунку 2.1.

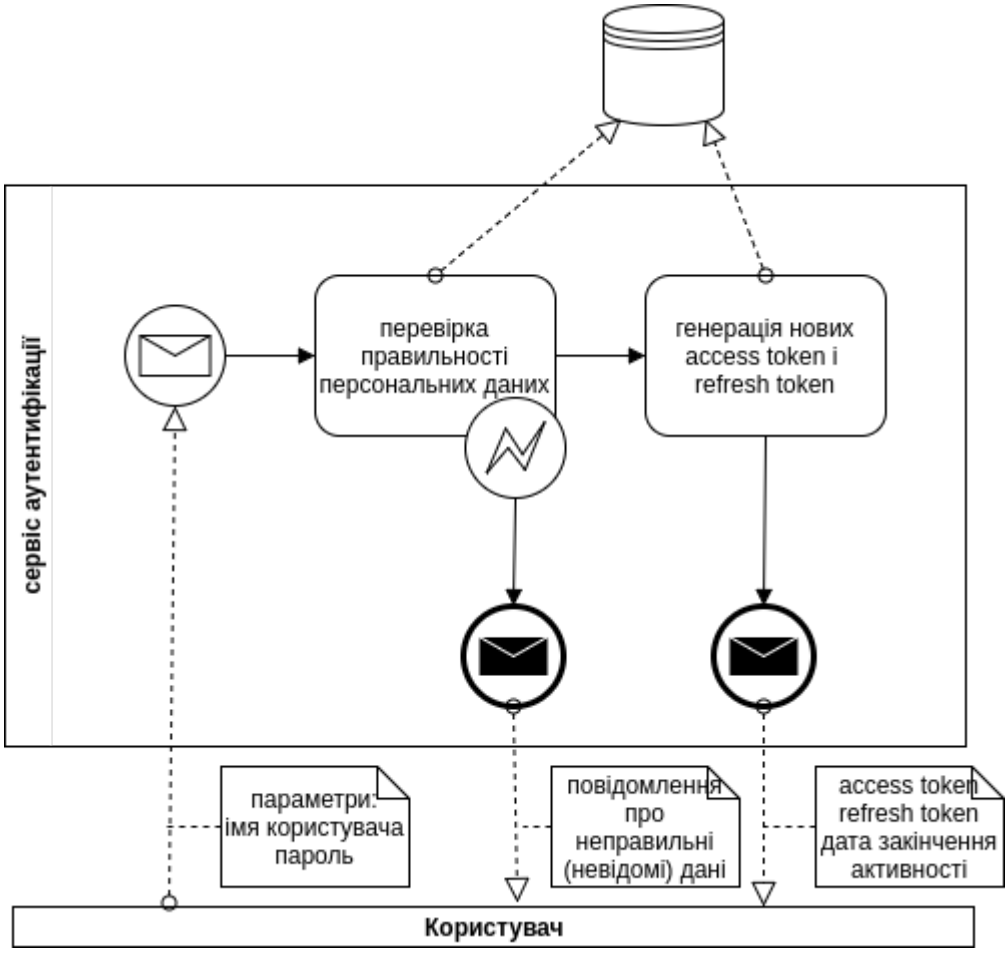


Рисунок 2.1 - Генерація токену

У випадку коли користувач має токен з завершеним терміном дії, але з активним токеном оновлення, то він може отримати новий токен доступу та токен оновлення. Опис процесу показано на рисунку 2.2.

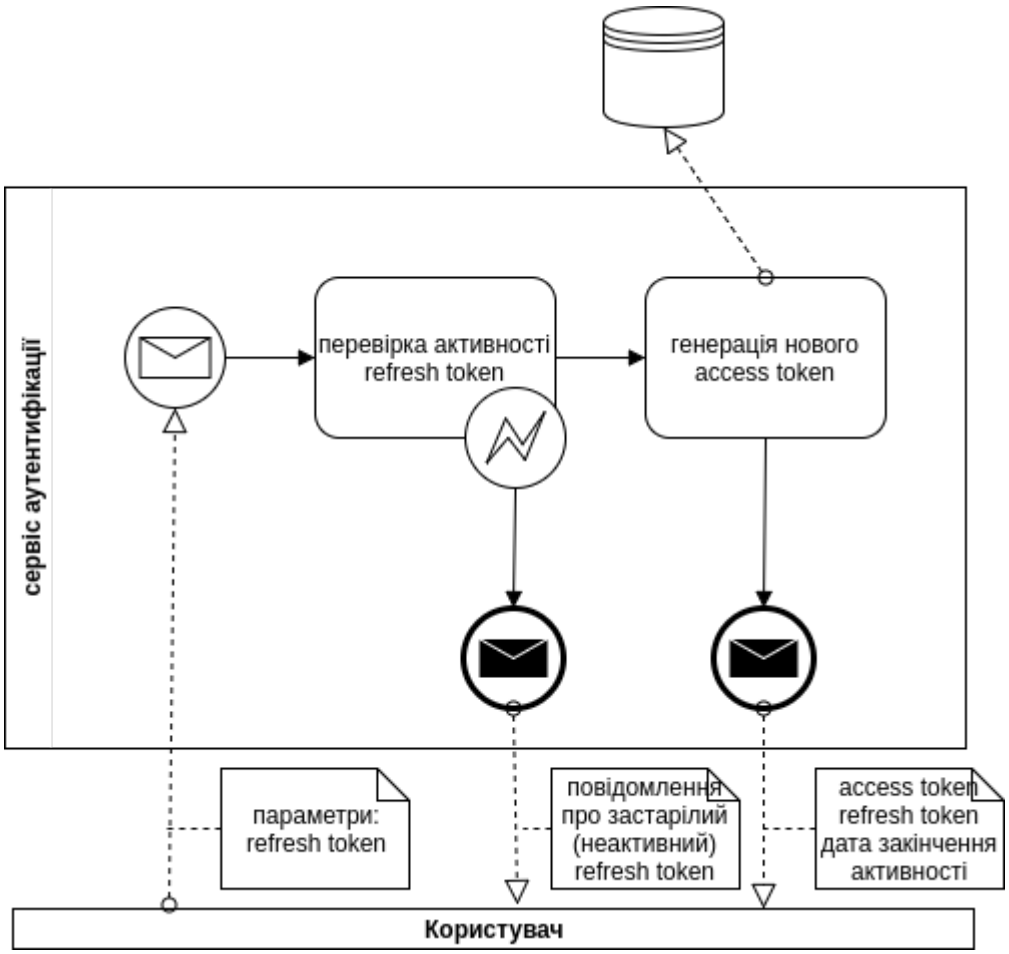


Рисунок 2.2 - Оновлення токена

Після успішного отримання токена користувач повинен відправляти його в заголовках запиту.

2.2 Архітектура програмного забезпечення

Під час аналізу можливої архітектури програмного забезпечення було обрано монолітну архітектуру. Оскільки у проекті тільки три основні частини, не має сенсу розбивати їх на мікросервіси. Причому усі частини тісно зв’язані між собою, тож зручніше буде працювати з ними всіма разом.

Структура БД наведена у документі «Структура БД»

2.2.1 API

Для реалізації API було використано підхід до архітектури REST.

REST (Representational state transfer) - це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови веб-служб.

У загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх прошарків. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожен URL в свою чергу має строго заданий формат.

Відсутність додаткових внутрішніх прошарків означає передачу даних в тому ж вигляді, що і самі дані. Тобто ми не загортаємо дані в XML, як це робить SOAP і XML-RPC, не використовуємо AMF, як це робить Flash і т.д.

Кожна одиниця інформації однозначно визначається URL - це значить, що URL по суті є первинним ключем для одиниці даних. Тобто наприклад третій користувач зі списку користувачіві матиме вигляд /user/3/.

Як відбувається управління інформацією сервісу - це цілком і повністю ґрунтується на протоколі передачі даних. Найбільш поширений протокол звичайно ж HTTP. Так ось, для HTTP ді] над даними зада.ться за допомогою методів: GET (отримати), PUT (замінити), POST (додати), DELETE (видалити), PATCH (частково змінити).

Ось як це буде виглядати на прикладі:

GET / users / - отримати список всіх користувачів

GET / users / 3 / - отримати користувача з ID 3

PUT / users / 3 / - змінити коритувача

DELETE / users / 3 - видалити користувача

У таблиці 2.1 описано доступні методи API:

Таблиця 2.1 - Доступні методи API

№	Назва метода	Опис метода
1	authorize	Реалізовує функціонал надавання користувачем доступу додатку для отримання персональних даних. Використовується для наступних випадків: authorization code та implicit grant.
2	token	Реалізовує логіку отримання та оновлення токенів доступу. Використовується для наступних випадків: authorization code, password, client credentials.
3	revoke_token	Реалізовує логіку видалення токена доступу або токена оновлення.
4	introspect	Реалізовує логіку перевірки токена на активність та повертає всю інформацію щодо нього. Для доступу до даного методу потрібно передати токен доступу в заголовках запиту, у якого є дозвіл на таку перевірку.

У таблиці 2.2 описано формат даних, які необхідно відправляти на відповідні методи.

Таблиця 2.2 - опис формату даних

№	Назва метода	Структура даних
1	authorize	У GET параметрах потрібно вказати response_type та client_id. Приклад: /o/authorize/?response_type=code&state= random_state_string&client_id= NOb0Ssfxw77NsFexj4fZSICSaQStV3XAHqs8P3Nl

продовження таблиці 2.2

2	token	У POST параметрах потрібно вказати code, який було отримано на попередньому кроці, а також redirect_uri та grant_type. Приклад:
		1. code:
		MHvdTYNdHQuks6Gj4564m32NKnFyxa
		2. redirect_uri:
		http://localhost:8000/callback_uri/
		3. grant_type:
		authorization_code
		
		У випадку передачі логіну та пароля користувача напряму потрібно передати наступні параметри:
		{
		'grant_type': 'password',
		'username': ‘username’,
		'password': ‘qwerty123’,
		'client_id': ‘mgI89234NFJDfjd8fjdsU’,
		'client_secret': ‘zb3U0OaU2xoM9FWZ4ioEQ78KYDFQUm9’
		}

3	revoke_token	{ "token": "3102jfdsifnsdif",

		"client_id": "NOb0Ssfxw77NsFexj4fZSICSaQStV3XAHqs8P3Nl" }

продовження таблиці 2.2

4	introspect	{ "token": "3102jfdsifnsdif" }

2.2.2 Адміністративна панель

Адміністративна панель повинна задовольняти мінімальні потреби редагування даних, а саме виконувати CRUD операції над всіма таблицями БД. Для цього буде використано Django-admin, яка має зручний та зрозумілий інтерфейс, та який адаптовано під різні пристрої.

2.3 Конструювання програмного забезпечення

2.3.1 Основна мова програмування

Основною мовою програмування було обрано Python. Python сьогодні в лідерах мов, що використовуються для програмування серверної частини сайт. Через це, існує багато різниї бібліотек и фреймворків.

Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за C обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу.

- типи даних високого рівня дозволяють виразити складні операції однією інструкцією;
- групування інструкцій виконується за допомогою відступів замість фігурних дужок;
- немає необхідності в оголошенні змінних;
- керування пам'яттю - цілком автоматичне — не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози “небезпечного посилання”.
- типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив може містити об'єкти різних типів. Традиційні мови не надають такої можливості.
- операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

2.3.2 Фреймворк розробки

Фреймворком для розробки веб-застосунку було обрано Django. Стороння бібліотека Django REST Framework дозволяє ефективно та швидко створювати REST API у проекті.

Що таке Django?

Django - це високорівневий веб-фреймворк Python. Він допомагає створювати веб-сайти з легкістю. Django піклується про більшість шаблонних рішень у веб розробці. Це безкоштовний і відкритий веб-фреймворк з відмінною документацією.

Переваги Django:

1. Безпечний

Django допомагає розробникам уникнути багатьох помилок безпеки. Наприклад, Django забезпечує безпечний спосіб керування обліковими записами користувачів і паролями.

Django забезпечує захист від більшості вразливостей сайту за замовчуванням. Вона включає в себе SQL Injection, XSS, CSRF, clickjacking і навіть більше.

2. Універсальний

Django може використовуватися для створення практично будь-якого типу веб-сайтів, таких як управління контентом, сайти новин, сайти електронної комерції, сайти соціальних мереж, вікі та ін. Ви можете створити будь-який сайт за допомогою Django з меншими витратами.

3. Портативний

Django написаний на Python. Вам не доведеться турбуватися про платформу під час розробки додатків у Django. Python працює на будь-якій платформі, такі як Linux, Windows, Mac OS X.

4. Активно підтримується розробниками

Код Django написаний на принципах, які спонукають користувачів створювати підтримуваний і багаторазовий код. Django зменшує кількість шаблонного коду у проекті и надає його «з коробки», наприклад, система адміністрування або міграції БД.

Архітектура застосунку Django наведено на рисунку 2.3.

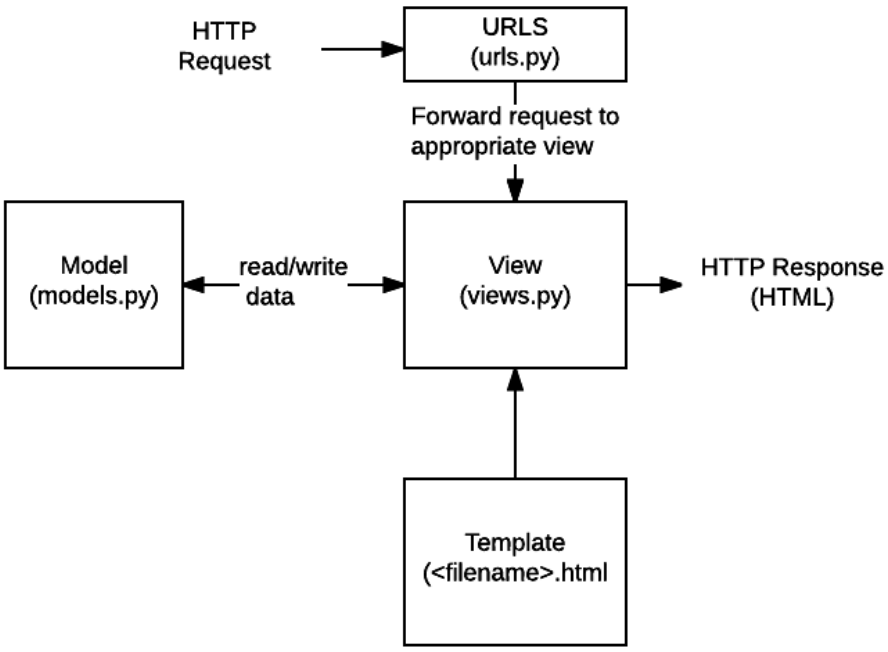


Рисунок 2.3 - архітектура застосунку Django

URLs

URL маппер використовується для перенаправлення запитів HTTP на відповідну view на основі URL-адреси запиту. Він також відповідає конкретним паттернам рядків або цифр, які з'являються в URL-адресі, і передає їх у view як дані.

View

View - це функція обробки запитів, яка приймає HTTP-запити і повертає відповіді HTTP. View отримують доступ до даних, необхідних для задоволення запитів за допомогою моделей і форматування відповідей у шаблони.

Model

Моделі - це об'єкти Python, які визначають структури даних програми. Вони забезпечують керуванням (створення, оновлення, читання, видалення) даних програми в базі даних.

Templates

Template - це текстовий файл, який визначає макет файлу. View може динамічно створювати HTML-сторінку за допомогою HTML-шаблону, заповнюючи його даними моделі. Шаблон може використовуватися для визначення структури будь-якого типу файлу; він не обов’язково повинен бути формату HTML.

2.4 Аналіз безпеки даних

У фреймворк Django вбудовано велику кількість впроваджень щодо підвищення безпеки даних, а також унеможливлення певних атак.

Django має захист проти:

- XSS – дозволяє встроювати скрипти у браузері інших користувачів
- CSRF attacks - дозволяють зловмиснику виконувати дії, використовуючи облікові дані іншого користувача без відома або згоди цього користувача.
- SQL-ін’єкцій
- Clickjacking

Для зберігання паролів користувачів використовується алгоритм PBKDF2. Пароль збегірається у форматі:

<algorithm>$<iterations>$<salt>$<hash>

SSL/HTTPS

Заради безпеки завжди краще розгортати свій сайт на HTTPS. Без цього зловмиснам можна присвоїти аутентифікаційні дані або будь-яку іншу інформацію, передану між клієнтом і сервером, а в деяких випадках - змінити дані, які надсилаються в будь-якому напрямку.

2.5 Висновки по розділу

У даному розділі було проаналізовано бізнес-процеси програмного забезпечення. Була визначена структура БД, а також обрана мова програмування та допоміжні фреймворки.

Було визначено які методи API повинні бути реалізовані, а також структура даних, що будуть надходити.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для забезпечення безперебійної роботи програми було проведено ряд тестувань, щоб віднайти і виправити усі знайдені неполадки програми. В ході перевіри було використано такі підходи до тестування:

- функціональне тестування;
- димне тестування;
- стрес тестування;
- тестування інтерфейсу;
- тестування зручності використання.

3.2 Опис процесів тестування

Детальний опис процесів тестування представлено у додатку В.

3.3 Опис контрольного прикладу

Детальний опис контрольного прикладу представлено у додатку В.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання ПЗ потрібен комп’ютер з будь-якою операційною системою. По-перше, повинен бути встановлений Python версії > 3.6. Також через менеджер пакетів pip потрібно встановити Django та інші пакети. Для локального запуску проекту цього буде достатьо.

Для того, щоб розгорнути проект на production, потрібно налаштувати gunicorn та nginx. Для підвищення безпеки краще впровадити підтримку HTTPS запитів, а усі HTTP запити перенаправляти на HTTPS.

4.2 Робота з програмним забезпеченням

Детальну інструкцію роботи із клієнтською частиною програмного забезпечення наведено у додатку “Керівництво користувача”.

ВИСНОВКИ

Під час розробки дипломного проекту було, по-перше, ретельно проаналізовано предметну область, а саме – доступні технічні рішення та програмні продукти.

Було розроблене програмне забезпечення, яке складається з 3 частин: API, адміністративна частина та графічний інтерфейс для керування додатками у контексті OAuth2. Після впровадження даної розробки підвищиться як швидкодія роботи, так і безпека даних користувачів. Інтегрувати сторонні додатки стане дуже просто и швидко, що однозначно є великим плюсом для проекту та компанії, і може відкрити нові можливості для росту.

Дане рішення може працювати і як окремний мікросервіс, і як частина монолітного проекту. Було проведено тестування та виправлення помилок.

Було розроблено технічну документацію, схеми бізнес-процесів, варіанти використання, а також керівництво для користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Обзор способов и протоколов аутентификации в веб-приложениях [Електронний ресурс]: (Стаття) / habr.com . – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://habr.com/ru/company/dataart/blog/262817/ . – Назва з екрана.

2. Guide to an OAuth2 API with Django. [Електронний ресурс]: (Стаття) / medium.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://medium.com/@halfspring/guide-to-an-oauth2-api-with-django-6ba66a31d6d . – Назва з екрана.

3. Введение в OAuth 2. [Електронний ресурс]: (Стаття) / digitalocean.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://www.digitalocean.com/community/tutorials/oauth-2-ru . – Назва з екрана.

4. Best identity management software of 2019. [Електронний ресурс]: (Стаття) / techradar.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://www.techradar.com/best/best-identity-management-software . – Назва з екрана.

5. 9-vendor authentication roundup: The good, the bad and the ugly. [Електронний ресурс]: (Стаття) / networkworld.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://www.networkworld.com/article/3077843/9-vendor-authentication-roundup-the-good-the-bad-and-the-ugly.html . – Назва з екрана.

6. How We Solved Authentication and Authorization in Our Microservice. [Електронний ресурс]: (Стаття) / medium.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https:// medium.com/technology-learning/how-we-solved-authentication-and-authorization-in-our-microservice-architecture-994539d1b6e6 – Назва з екрана.

7. Adolfo Eloy Nascimento OAuth 2.0 Cookbook: Protect your web applications using Spring Security, 2017. 420 c.

8. The OAuth 2.0 Authorization Framework: Bearer Token Usage. [Електронний ресурс]: (Стаття) / tools.ietf.org . – Електрон. дан. (1 файл). – 2012. – Режим доступу: https://tools.ietf.org/html/rfc6750 . – Назва з екрана.

9. OAuth 2.0 and the Road to Hell [Електронний ресурс]: (Стаття) / hueniverse.com . – Електрон. дан. (1 файл). – 2012. – Режим доступу: https://hueniverse.com/oauth-2-0-and-the-road-to-hell-8eec45921529 . – Назва з екрана.

10. OAuth Security Advisory: 2009.1. [Електронний ресурс]: (Стаття) / oauth.net . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https:// oauth.net/advisories/2009-1/ . – Назва з екрана.

11. GitHub - OlegKozlovskui/Security-tutorial. [Електронний ресурс]: (Стаття) / github.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://github.com/OlegKozlovskui/Security-tutorial . – Назва з екрана.

12. Security-tutorial/README.md at master · OlegKozlovskui/Security-tutorial · GitHub. [Електронний ресурс]: (Стаття) / github.com . – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://github.com/OlegKozlovskui/Security-tutorial/blob/master/README.md . – Назва з екрана.

13. Що таке Python? [Електронний ресурс]: (Стаття) / plug.org.ua . – Електрон. дан. (1 файл). – 2012. – Режим доступу: http://www.plug.org.ua/documentation/about-python . – Назва з екрана.

14. Методичні вказівки до дипломного проекту з курсу Комп'ютерні науки, НТУУ «КПІ» [Електронний ресурс]: (Стаття) / ua.kursoviks.com.ua . – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://ua.kursoviks.com.ua/metodychni_vkazivky/article_post/1684-metodichni-vkazivki-do-diplomnogo-proyektu-z-kursu-komp-yuterni-nauki-ntuu-kpi . – Назва з екрана.

15. Анотація структура та обсяг роботи [Електронний ресурс]: (Стаття) / refdb.ru . – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://refdb.ru/look/2533440-pall.html . – Назва з екрана.

16. Тезисы_REST. Архітектура rest що таке rst [Електронний ресурс]: (Стаття) / topuch.ru . – Електрон. дан. (1 файл). – 2019. – Режим доступу: http://topuch.ru/arhitektura-rest-sho-take-rst/index.html . – Назва з екрана.

17. Розробка web-додатку для онлайн реєстрації пацієнтів на прийом до лікаря [Електронний ресурс]: (Стаття) / inmad.vntu.edu.ua . – Електрон. дан. (1 файл). – 2019. – Режим доступу: http://inmad.vntu.edu.ua/portal/static/84DDB2D1-CA67-4EF9-A4F5-D5A72B79D470.pdf . – Назва з екрана.

18. OAuth 2 Simplified [Електронний ресурс]: (Стаття) / aaronparecki.com . – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://aaronparecki.com/oauth-2-simplified/ . – Назва з екрана.

19. ЗАХИСТ ДАНИХ У WEB-ДОДАТКАХ [Електронний ресурс]: (Стаття) / rusnauka.com . – Електрон. дан. (1 файл). – 2019. – Режим доступу: http://www.rusnauka.com/22_AND_2016/Informatica/4_215343.doc.htm . – Назва з екрана.

20. Програмне забезпечення [Електронний ресурс]: (Стаття) / interdip.com.ua . – Електрон. дан. (1 файл). – 2019. – Режим доступу: http://interdip.com.ua/uk/gotoviye-raboty.html?page=shop.product_details&flypage=zakaz.tpl&product_id=1656&category_id=261 . – Назва з екрана.

21. ОФОРМЛЕННЯ ТЕКСТОВИХ ДОКУМЕНТІВ В ПРОЕКТАХ ТА РОБОТАХ [Електронний ресурс]: (Стаття) / aesiitf.kpi.ua . – Електрон. дан. (1 файл). – 2019. – Режим доступу: http://aesiitf.kpi.ua/wp-content/uploads/2015/06/oformlennya_tekstovih_dokumentiv_v_proektah_ta_robotah.pdf – Назва з екрана.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

СЕРВІС АВТОРИЗАЦІЇ ЗА ДОПОМОГОЮ СТАНДАРТУ OAUTH2

Опис програми

КПІ.ІП-5116.045490.02.13

“ПОГОДЖЕНО”

Керівник проекту:

_____ О. Д. Фіногенов

Виконавець:

_____В. В. Патерило

Нормоконтроль:

_____ М.М. Головченко

Київ – 2019 року

Тексти програмного коду

Сервіс авторизації за допомогою стандарту OAuth2

(Найменування програми (документа))

CD-R

(Вид носія даних)

19 арк, 228 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2019

```
import re
from urllib.parse import urlsplit

from django.core.exceptions import ValidationError
from django.core.validators import URLValidator
from django.utils.encoding import force_text


class URIValidator(URLValidator):
    scheme_re = r"^(?:[a-z][a-z0-9\.\-\+]*)://"

    dotless_domain_re = r"(?!-)[A-Z\d-]{1,63}(?<!-)"
    host_re = "|".join((
        r"(?:" + URLValidator.host_re,
        URLValidator.ipv4_re,
        URLValidator.ipv6_re,
        dotless_domain_re + ")"
    ))
    port_re = r"(?::\d{2,5})?"
    path_re = r"(?:[/?#][^\s]*)?"
    regex = re.compile(scheme_re + host_re + port_re + path_re,
re.IGNORECASE)


class RedirectURIValidator(URIValidator):
    def __init__(self, allowed_schemes, allow_fragments=False):
        super().__init__(schemes=allowed_schemes)
        self.allow_fragments = allow_fragments
```

```
        def __call__(self, value):
            super().__call__(value)
            value = force_text(value)
            scheme, netloc, path, query, fragment = urlsplit(value)
            if fragment and not self.allow_fragments:
                raise ValidationError("Redirect URIs must not contain fragments")


    ##
    # WildcardSet is a special set that contains everything.
    # This is required in order to move validation of the scheme from
    # URLValidator (the base class of URIValidator), to
OAuth2Application.clean().

    class WildcardSet(set):
        """
        A set that always returns True on `in`.
        """
        def __contains__(self, item):
            return True

    from django.conf.urls import url

    from . import views


    app_name = "oauth2_provider"
```

```
    base_urlpatterns = [
        url(r"^authorize/$", views.AuthorizationView.as_view(), name="authorize"),
        url(r"^token/$", views.TokenView.as_view(), name="token"),
        url(r"^revoke_token/$", views.RevokeTokenView.as_view(),
name="revoke-token"),
        url(r"^introspect/$", views.IntrospectTokenView.as_view(),
name="introspect"),
    ]


    management_urlpatterns = [
        # Application management views
        url(r"^applications/$", views.ApplicationList.as_view(), name="list"),
        url(r"^applications/register/$", views.ApplicationRegistration.as_view(),
name="register"),
        url(r"^applications/(?P<pk>[\w-]+)/$", views.ApplicationDetail.as_view(),
name="detail"),
        url(r"^applications/(?P<pk>[\w-]+)/delete/$",
views.ApplicationDelete.as_view(), name="delete"),
        url(r"^applications/(?P<pk>[\w-]+)/update/$",
views.ApplicationUpdate.as_view(), name="update"),
        # Token management views
        url(r"^authorized_tokens/$", views.AuthorizedTokensListView.as_view(),
name="authorized-token-list"),
        url(r"^authorized_tokens/(?P<pk>[\w-]+)/delete/$",
views.AuthorizedTokenDeleteView.as_view(),
            name="authorized-token-delete"),
    ]
```

```
urlpatterns = base_urlpatterns + management_urlpatterns

from .settings import oauth2_settings


class BaseScopes(object):
    def get_all_scopes(self):
        """
        Return a dict-like object with all the scopes available in the
        system. The key should be the scope name and the value should be
        the description.

        ex: {"read": "A read scope", "write": "A write scope"}
        """
        raise NotImplementedError("")

    def get_available_scopes(self, application=None, request=None, *args, **kwargs):
        """
        Return a list of scopes available for the current application/request.

        TODO: add info on where and why this method is called.

        ex: ["read", "write"]
        """
        raise NotImplementedError("")
```

```
        def get_default_scopes(self, application=None, request=None, *args,
**kwargs):
            """
            Return a list of the default scopes for the current application/request.
            This MUST be a subset of the scopes returned by `get_available_scopes`.

            TODO: add info on where and why this method is called.

            ex: ["read"]
            """
            raise NotImplementedError("")


    class SettingsScopes(BaseScopes):
        def get_all_scopes(self):
            return oauth2_settings.SCOPES

        def get_available_scopes(self, application=None, request=None, *args,
**kwargs):
            return oauth2_settings._SCOPES

        def get_default_scopes(self, application=None, request=None, *args,
**kwargs):
            return oauth2_settings._DEFAULT_SCOPES


    def get_scopes_backend():
        scopes_class = oauth2_settings.SCOPES_BACKEND_CLASS
        return scopes_class()
```

```
import base64
import binascii
import logging
from collections import OrderedDict
from datetime import datetime, timedelta
from urllib.parse import unquote_plus

import requests
from django.conf import settings
from django.contrib.auth import authenticate, get_user_model
from django.core.exceptions import ObjectDoesNotExist
from django.db import transaction
from django.db.models import Q
from django.utils import timezone
from django.utils.timezone import make_aware
from django.utils.translation import ugettext_lazy as _
from oauthlib.oauth2 import RequestValidator

from .exceptions import FatalClientError
from .models import (
    AbstractApplication, get_access_token_model,
    get_application_model, get_grant_model, get_refresh_token_model
)
from .scopes import get_scopes_backend
from .settings import oauth2_settings
```

```
log = logging.getLogger("oauth2_provider")

GRANT_TYPE_MAPPING = {
    "authorization_code":
(AbstractApplication.GRANT_AUTHORIZATION_CODE, ),
    "password": (AbstractApplication.GRANT_PASSWORD, ),
    "client_credentials":
(AbstractApplication.GRANT_CLIENT_CREDENTIALS, ),
    "refresh_token": (
        AbstractApplication.GRANT_AUTHORIZATION_CODE,
        AbstractApplication.GRANT_PASSWORD,
        AbstractApplication.GRANT_CLIENT_CREDENTIALS,
    )
}

Application = get_application_model()
AccessToken = get_access_token_model()
Grant = get_grant_model()
RefreshToken = get_refresh_token_model()
UserModel = get_user_model()



class OAuth2Validator(RequestValidator):
    def _extract_basic_auth(self, request):
        """
        Return authentication string if request contains basic auth credentials,
        otherwise return None
        """
        auth = request.headers.get("HTTP_AUTHORIZATION", None)
```

```
            if not auth:
                return None

            splitted = auth.split(" ", 1)
            if len(splitted) != 2:
                return None
            auth_type, auth_string = splitted

            if auth_type != "Basic":
                return None

            return auth_string

        def _authenticate_basic_auth(self, request):
            """
            Authenticates with HTTP Basic Auth.

            client_id and client_secret must be encoded with
            "application/x-www-form-urlencoded" encoding algorithm.
            """
            auth_string = self._extract_basic_auth(request)
            if not auth_string:
                return False

            try:
                encoding = request.encoding or settings.DEFAULT_CHARSET or "utf-
8"
            except AttributeError:
                encoding = "utf-8"
```

```
            try:
                b64_decoded = base64.b64decode(auth_string)
            except (TypeError, binascii.Error):
                log.debug("Failed basic auth: %r can't be decoded as base64",
auth_string)
                return False

            try:
                auth_string_decoded = b64_decoded.decode(encoding)
            except UnicodeDecodeError:
                log.debug(
                    "Failed basic auth: %r can't be decoded as unicode by %r",
                    auth_string, encoding
                )
                return False

            try:
                client_id, client_secret = map(unquote_plus,
auth_string_decoded.split(":", 1))
            except ValueError:
                log.debug("Failed basic auth, Invalid base64 encoding.")
                return False

            if self._load_application(client_id, request) is None:
                log.debug("Failed basic auth: Application %s does not exist" %
client_id)
                return False
            elif request.client.client_id != client_id:
```

```
            log.debug("Failed basic auth: wrong client id %s" % client_id)
            return False
        elif request.client.client_secret != client_secret:
            log.debug("Failed basic auth: wrong client secret %s" % client_secret)
            return False
        else:
            return True

    def _authenticate_request_body(self, request):
        """
        Try to authenticate the client using client_id and client_secret
        parameters included in body.
        """
        # TODO: check if oauthlib has already unquoted client_id and client_secret
        try:
            client_id = request.client_id
            client_secret = request.client_secret
        except AttributeError:
            return False

        if self._load_application(client_id, request) is None:
            log.debug("Failed body auth: Application %s does not exists" %
client_id)
            return False
        elif request.client.client_secret != client_secret:
            log.debug("Failed body auth: wrong client secret %s" % client_secret)
            return False
        else:
            return True
```

```
    def _load_application(self, client_id, request):
        """
        If request.client was not set, load application instance for given
        client_id and store it in request.client
        """

        # we want to be sure that request has the client attribute!
        assert hasattr(request, "client"), '"request" instance has no "client" attribute'

        try:
            request.client = request.client or Application.objects.get(client_id=client_id)
            # Check that the application can be used (defaults to always True)
            if not request.client.is_usable(request):
                log.debug("Failed body authentication: Application %r is disabled" % (client_id))
                return None
            return request.client
        except Application.DoesNotExist:
            log.debug("Failed body authentication: Application %r does not exist" % (client_id))
            return None

    def _set_oauth2_error_on_request(self, request, access_token, scopes):
        if access_token is None:
            error = OrderedDict([
                ("error", "invalid_token", ),
                ("error_description", _("The access token is invalid."), ),
```

```
            ])
        elif access_token.is_expired():
            error = OrderedDict([
                ("error", "invalid_token", ),
                ("error_description", _("The access token has expired."), ),
            ])
        elif not access_token.allow_scopes(scopes):
            error = OrderedDict([
                ("error", "insufficient_scope", ),
                ("error_description", _("The access token is valid but does not have enough scope."), ),
            ])
        else:
            log.warning("OAuth2 access token is invalid for an unknown reason.")
            error = OrderedDict([
                ("error", "invalid_token", ),
            ])
        request.oauth2_error = error
        return request

    def client_authentication_required(self, request, *args, **kwargs):
        """
        Determine if the client has to be authenticated

        This method is called only for grant types that supports client authentication:
            * Authorization code grant
            * Resource owner password grant
            * Refresh token grant
```

```
        """
        if self._extract_basic_auth(request):
            return True

        try:
            if request.client_id and request.client_secret:
                return True
        except AttributeError:
            log.debug("Client ID or client secret not provided...")
            pass

        self._load_application(request.client_id, request)
        if request.client:
            return request.client.client_type ==
AbstractApplication.CLIENT_CONFIDENTIAL

        return super().client_authentication_required(request, *args, **kwargs)

    def authenticate_client(self, request, *args, **kwargs):
        """
        Check if client exists and is authenticating itself
        """
        authenticated = self._authenticate_basic_auth(request)

        if not authenticated:
            authenticated = self._authenticate_request_body(request)

        return authenticated
```

```
    def authenticate_client_id(self, client_id, request, *args, **kwargs):
        if self._load_application(client_id, request) is not None:
            log.debug("Application %r has type %r" % (client_id, request.client.client_type))
            return request.client.client_type != AbstractApplication.CLIENT_CONFIDENTIAL
        return False

    def confirm_redirect_uri(self, client_id, code, redirect_uri, client, *args, **kwargs):
        """
        Ensure the redirect_uri is listed in the Application instance redirect_uris field
        """
        grant = Grant.objects.get(code=code, application=client)
        return grant.redirect_uri_allowed(redirect_uri)

    def invalidate_authorization_code(self, client_id, code, request, *args, **kwargs):
        """
        Remove the temporary grant used to swap the authorization token
        """
        grant = Grant.objects.get(code=code, application=request.client)
        grant.delete()

    def validate_client_id(self, client_id, request, *args, **kwargs):
        """
        Ensure an Application exists with given client_id.
        If it exists, it's assigned to request.client.
```

```
            """
            return self._load_application(client_id, request) is not None

        def get_default_redirect_uri(self, client_id, request, *args, **kwargs):
            return request.client.default_redirect_uri

        def _get_token_from_authentication_server(
                self, token, introspection_url, introspection_token,
introspection_credentials
        ):
            """Use external introspection endpoint to "crack open" the token.
            :param introspection_url: introspection endpoint URL
            :param introspection_token: Bearer token
            :param introspection_credentials: Basic Auth credentials (id,secret)
            :return: :class:`models.AccessToken`
            """
            headers = None
            if introspection_token:
                headers = {"Authorization": "Bearer {}".format(introspection_token)}
            elif introspection_credentials:
                client_id = introspection_credentials[0].encode("utf-8")
                client_secret = introspection_credentials[1].encode("utf-8")
                basic_auth = base64.b64encode(client_id + b":" + client_secret)
                headers = {"Authorization": "Basic {}".format(basic_auth.decode("utf-
8"))}

            try:
                response = requests.post(
                    introspection_url,
```

```
                data={"token": token}, headers=headers
            )
        except requests.exceptions.RequestException:
            log.exception("Introspection: Failed POST to %r in token lookup",
introspection_url)
            return None

        try:
            content = response.json()
        except ValueError:
            log.exception("Introspection: Failed to parse response as json")
            return None

        if "active" in content and content["active"] is True:
            if "username" in content:
                user, _created = UserModel.objects.get_or_create(
                    **{UserModel.USERNAME_FIELD: content["username"]}
                )
            else:
                user = None

            max_caching_time = datetime.now() + timedelta(

seconds=oauth2_settings.RESOURCE_SERVER_TOKEN_CACHING_SECONDS
            )

            if "exp" in content:
                expires = datetime.utcfromtimestamp(content["exp"])
                if expires > max_caching_time:
```

```
            expires = max_caching_time
        else:
            expires = max_caching_time

        scope = content.get("scope", "")
        expires = make_aware(expires)

        access_token, _created = AccessToken.objects.update_or_create(
            token=token,
            defaults={
                "user": user,
                "application": None,
                "scope": scope,
                "expires": expires,
            })

        return access_token

    def validate_bearer_token(self, token, scopes, request):
        """
        When users try to access resources, check that provided token is valid
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Сервіс авторизації за допомогою стандарту OAuth2

Технічне завдання

КПІ.ІП-5116.045490.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____О. Д. Фіногенов

Нормоконтроль:

_____ М.М. Головченко

Виконавець:

_____В.В.Патерило

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ 3

2 ПІДСТАВА ДЛЯ РОЗРОБКИ 3

3 ПРИЗНАЧЕННЯ РОЗРОБКИ 5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6

4.1 Вимоги до функціональних характеристик 6

4.2 Вимоги до надійності 6

4.3 Умови експлуатації 6

4.4 Вимоги до складу і параметрів технічних засобів 7

4.5 Вимоги до інформаційної та програмної сумісності 7

4.6 Вимоги до маркування та пакування 7

4.7 Вимоги до транспортування та зберігання 7

4.8 Спеціальні вимоги 7

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ 8

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ 9

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ 11

7.1 Види випробувань 11

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Сервіс авторизації за допомогою стандарту OAuth2

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення «Сервіс авторизації за допомогою стандарту OAuth2», котре використовується для безпечного та зручного опрацювання даних користувачів та призначена для усіх програмних продуктів, в яких наявні дані користувачів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для тестування знань з обраної області з підтримкою ігрової механіки змагань є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання усіма програмними продуктами, у яких наявні користувачі.

Метою розробки є підвищення безпеки користувачів, зручний та уніфікований процес авторизації. Підвищення ефективності розробки в цілому – а саме інтегрування різних мікросервісів з основним проектом. Надалі для отримання доступу мікросервісами до даних про користувача буде використано тільки одну точка входу. Створення сервісу для авторизації користувачів, який буде складатися з API для керування користувачами, певного графічного інтерфесу, а також адміністративної панелі. В результаті використанні готового рішення будуть значно знижені ресурси на розробку ПЗ та підвищена продуктивність розробників. Головне – стандарт OAuth2 має доволі високий рівень безпеки, тож персональні дані користувачів будуть захищені від несанкціонованого доступу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для розробника:

- отримання та керування токенами користувачів;
- зручна інтеграція сторонніх сервісів
- графічний інтерфейс

4.1.1.2 Для адміністратора системи:

- Адміністративна панель

4.1.2 Розробку виконати на платформі Windows

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації

4.2.2 Передбачити захист від некоректних дій користувача

4.2.3 Забезпечити цілісність інформації в вхідних файлах

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96

4.3.2 Обслуговування

4.3.3 Обслуговуючий персонал

Не потребує спеціальних кваліфікацій для обслуговування.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах

4.4.2 Мінімальна конфігурація технічних засобів:

Тип процесору	Pentium.
Об‘єм ОЗП	512 Мб.
Об’єм відео пам’яті	256 Мб.
Об’єм фізичної пам’яті	250 Мб.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.)

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: JSON

4.5.3 Результати повинні бути представлені в наступному форматі: JSON

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків 5.3.2 - 5.3.6)

5.3.2 Технічне завдання

5.3.3 Керівництво користувача

5.3.4 Керівництво системного програміста

5.3.5 Керівництво адміністратора

5.3.6 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на листі формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема БД

5.4.2 Схема структурна класів програмного забезпечення

5.4.3 Креслення вигляду екранних форм

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк,	Звітність
1	Вивчення літератури за тематикою проекту	20.04.19	
2	Розробка технічного завдання	23.04.19	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	25.04.19	Специфікації програмного забезпечення
4	Проектування структури програмного забезпечення, проектування компонентів	28.04.19	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5	Програмна реалізація програмного забезпечення	15.05.19	Тексти програмного забезпечення
6	Тестування програмного забезпечення	19.05.19	Тести, результати тестування
7	Розробка матеріалів текстової частини проекту	27.05.19	Пояснювальна записка.
8	Розробка матеріалів графічної частини проекту	07.06.19	Графічний матеріал проекту

9	Оформлення технічної документації проекту	07.06.19	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

СЕРВІС АВТОРИЗАЦІЇ ЗА ДОПОМОГОЮ СТАНДАРТУ OAUTH2

Програма та методика тестування

КПІ.ІП-5116.045490.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____ О. Д. Фіногенов

Виконавець:

_____В. В. Патерило

Нормоконтроль:

_____М.М.Головченко

Київ – 2019 року

ЗМІСТ

1. ОБ'ЄКТ ВИПРОБУВАНЬ..... 3

2. МЕТА ТЕСТУВАННЯ..... 4

3. МЕТОДИ ТЕСТУВАННЯ..... 5

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ 6

5. АНАЛІЗ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 7

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Сервіс авторизації за допомогою стандарту OAuth2, що включає в себе API, графічний інтерфейс та адміністративну панель.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок застосунку;
- наявність доступу до API;
- забезпечення належного рівня безпеки даних;
- можливість створення токена за допомогою API;
- можливість створення додатку через графічний інтерфейс;
- відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (базове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність застосунку перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування роботи застосунку при різних швидкостях інтернету;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

1 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз якості ПЗ

Аналіз якості розроблених програмних засобів є одним з ключових елементів процесу розробки та впровадження. Якість – суб'єктивне поняття, що позначає відповідність між очікуваною користувачем поведінкою та реальністю. Оскільки якість не є вимірюваною тестування як один з елементів контролю якості займається перевіркою методом порівняння чіткої специфікації з результатами роботи коду у кінцевому наборі тестів.

План тестування наводить набір функцій програмних засобів які будуть протестовані, а також типи тестів, необхідні ресурси, тощо.

План тестування включає виконання тестування всіх частин продукту, від процесу отримання токена до створення додатку.

1.2 Опис процесів тестування

1.2.1 Дані до тестів

Вхідними даними до тестування застосунку можуть бути токен доступу, токен оновлення, інформація щодо терміну дії токену та список дозволів цього токена.

Вхідними даними для перевірки графічного інтерфейсу керування додатками є список додатків, інформаціх щодо кожного з них.

1.2.2 Задачі тесту

Задачею будь-якого тесту є перевірка правильності роботи програми в умовах виконання тесту, а також виявлення потенційних помилок у роботі.

1.3 Опис контрольного прикладу

В процесі тестування була перевірена уся функціональність підсистеми. Послідовно були перевірені всі варіанти використання, результати представлені у відповідних таблицях:

- Можливість створення токена за допомогою логіна і пароля (таблиця 1.1);
- Можливість створення додатку (таблиця 1.2);
- Можливість редагування даних користувача (таблиця 1.3);

Таблиця 1.1 – Можливість створення токена за допомогою логіна і пароля

Мета тесту	Перевірка можливості створення токена за допомогою логіна і пароля
Початковий стан	Відкритий API клієнт.
Вхідні дані	Логін та пароль користувача
Схема проведення тесту	Відправити дані на сервер
Очікуваний результат	Створено токен доступу і токен оновлення
Стан програмного продукту після проведення випробувань	Створено токен доступу і токен оновлення, які можна переглянути на сторінці списку токенів.

Таблиця 1.2 – Перевірка можливості створення додатку

Мета тесту	Перевірка можливості створення додатку
Початковий стан	Відкрита сторінка створення додатку
Вхідні дані	Назва додатку, тип, адреса перенаправлення
Схема проведення тесту	Ввести вхідні дані та натиснути кнопку «Створити»
Очікуваний результат	У списку додатків наявний створений додаток

Продовження таблиці 1.2

Стан програмного продукту після проведення випробувань	Сторінку перенаправлено на сторінку списку додатків

Таблиця 1.3 – Можливість редагування даних користувача

Мета тесту	Перевірка можливості редагування даних користувача
Початковий стан	Відкрита адміністративна частина сайту, а саме сторінка користувача
Вхідні дані	Нові дані користувача
Схема проведення тесту	Ввести вхідні дані та натиснути кнопку «Зберегти»
Очікуваний результат	Дані користувача оновлено
Стан програмного продукту після проведення випробувань	Сторінка перезавантажилась, показано оновлені дані

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

СЕРВІС АВТОРИЗАЦІЇ ЗА ДОПОМОГОЮ СТАНДАРТУ OAUTH2

Керівництво користувача

КПІ.ІП-5116.045490.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ О. Д. Фіногенов

Виконавець:

_____В. В. Патерило

Нормоконтроль:

_____ М.М. Головченко

Київ – 2019 року

ЗМІСТ

1. АДМІНІСТРАТИВНА ПАНЕЛЬ 3

2. ГРАФІЧНИЙ ІНТЕРФЕЙС ДЛЯ РЕДАГУВАННЯ ДОДАТКІВ .. 5

3. API 7

1 АДМІНІСТРАТИВНА ПАНЕЛЬ

Після запуску застосунку користувачу представлено стартове вікно авторизації, що показано на рисунку 1.1.

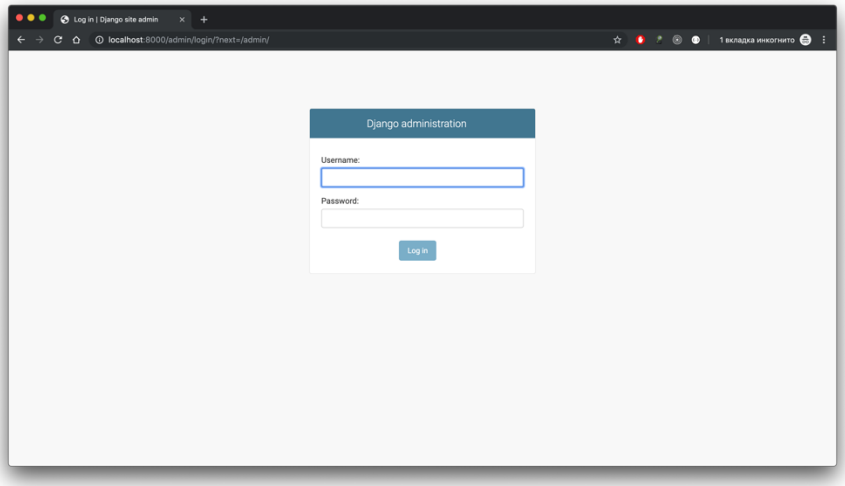


Рисунок 1.**Ошибка! Текст указанного стиля в документе отсутствует.**.1 - Авторизація до адміністративної панелі

Після введення правильного логіну та паролю та натисканню клавіші “SignIn” користувачу відобразиться адміністративна панель, що показано на рисунку 1.2.

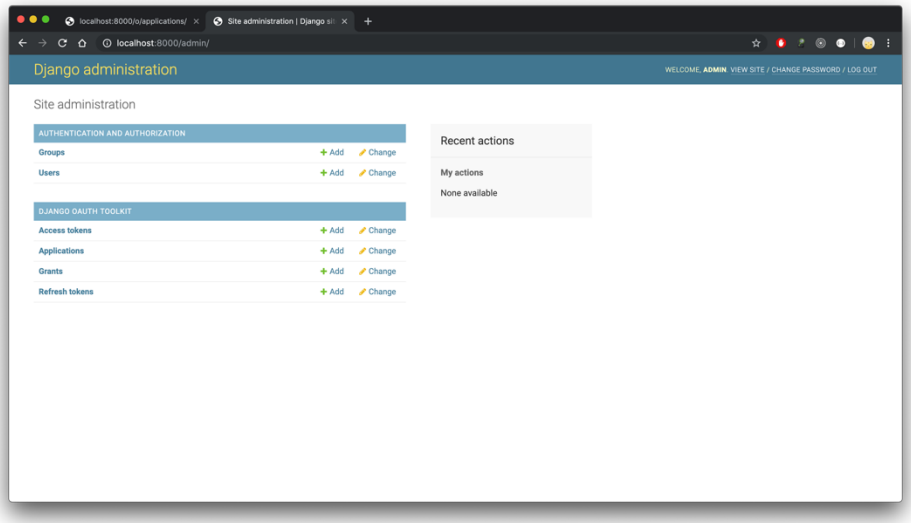


Рисунок 1.**Ошибка! Текст указанного стиля в документе отсутствует.**.2 - Адміністративна панель

Ми бачимо список відкритик до редагування таблиць. Оберемо, наприклад, таблицю Access Tokens. Для цього потрібно клікнути на назву таблиці і далі відбудеться перехід на сторінку списку токенів, як показано на рисунку 1.3.

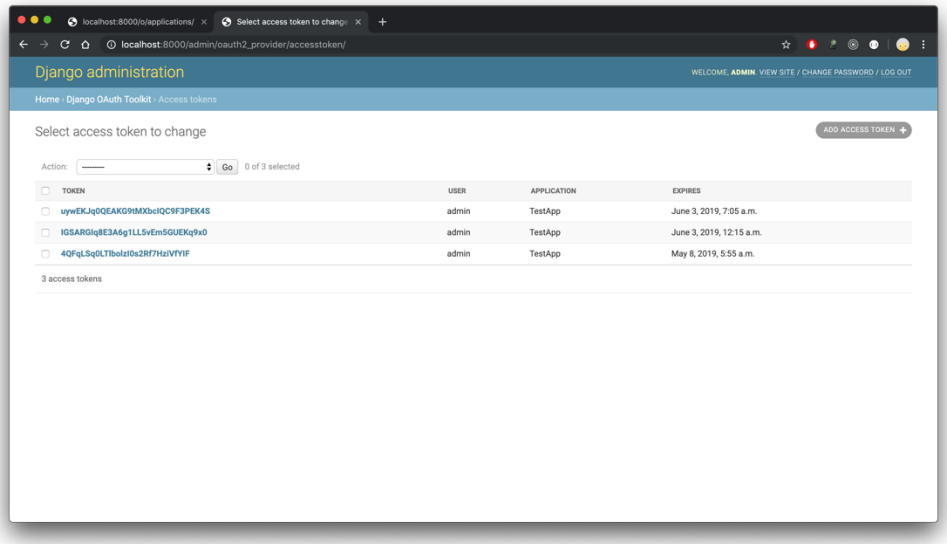


Рисунок 1.**Ошибка! Текст указанного стиля в документе отсутствует.**.3 - Перегляд списку об'єктів

Для того, щоб перегляну інформацію про конкретний об’єкт, потрібно натиснути на нього. Після цього буде здійснено перехід на сторінку об'єкта, як показано на рисунку 1.4.

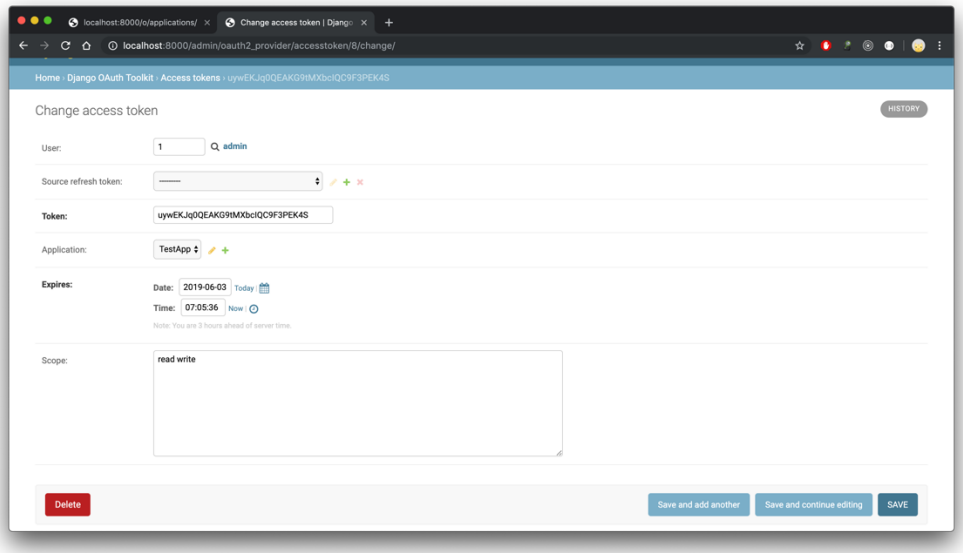


Рисунок 1.**Ошибка! Текст указанного стиля в документе отсутствует.**.4 - Перегляд даних токена

Після внесення змін потрібно натиснути кнопку «Зберегти».

2 ГРАФІЧНИЙ ІНТЕРФЕЙС ДЛЯ РЕДАГУВАННЯ ДОДАТКІВ

Здійснемо перехід на сторінку зі списком додатків, як зображено на рисунку 2.1.

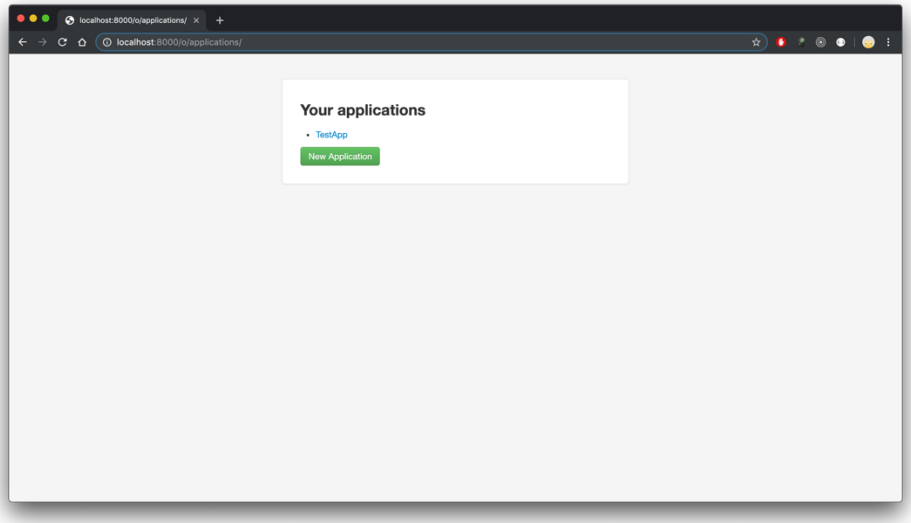


Рисунок 2.1 - Перегляд списку додатків

Після цього здійснемо перехід на сторінку додатку, щоб переглянути інформацію щодо нього. Потрібно клікнути на його назву, результат зображено на рисунку 2.2.

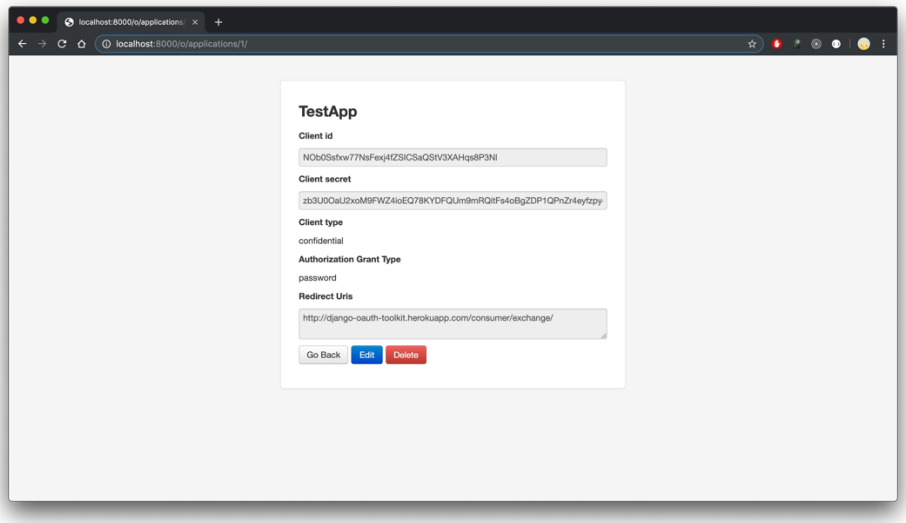


Рисунок 2.2 - Перегляд інформації щодо додатку

На цій сторінці можна видалити додаток та перейти на сторінку редагування. Для того щоб почати редагування, потрібно натиснути на кнопку Edit, результат показано на рисунку 2.3.

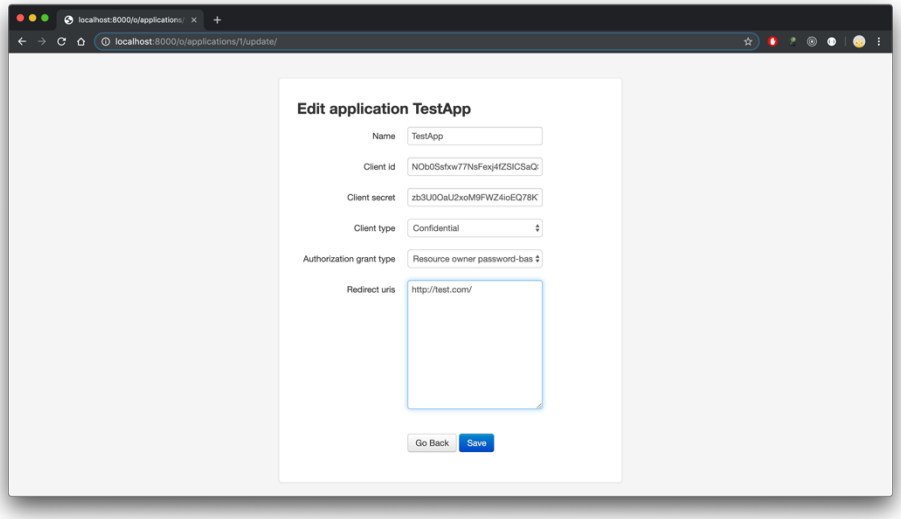


Рисунок 2.3 - Редагування даних додатку

Для того, щоб створити додаток, потрібно на сторінці зі списком додатків натиснути кнопку «Створити». Після цього буде здійснено перехід на сторінку створення додатку. Client_id та client_secret генеруються автоматично, інші дані потрібно ввести, як показано на рисунку 2.4.

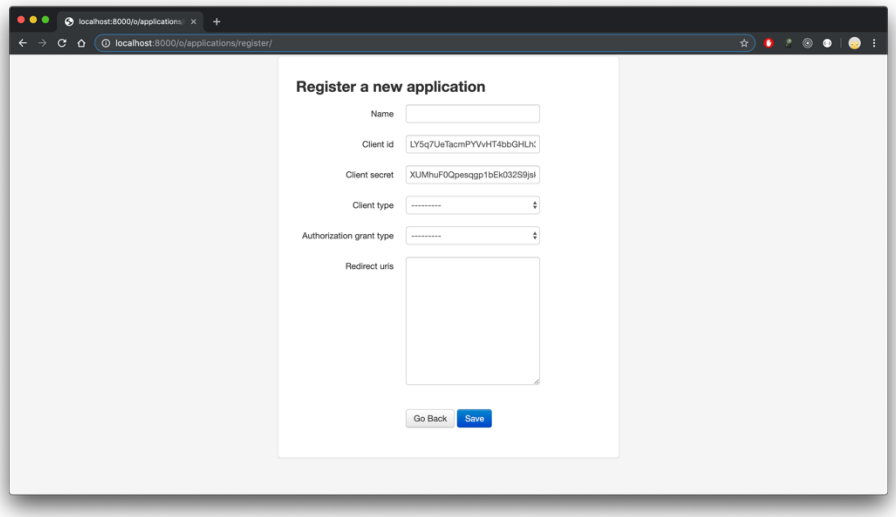


Рисунок 2.4 - Створення додатку

3 API

Для того, щоб отримати токен доступу за допомогою логіна і пароля користувача, необхідно відправити запит на url /o/token/ з даними, вказаними на рисунку 3.1.

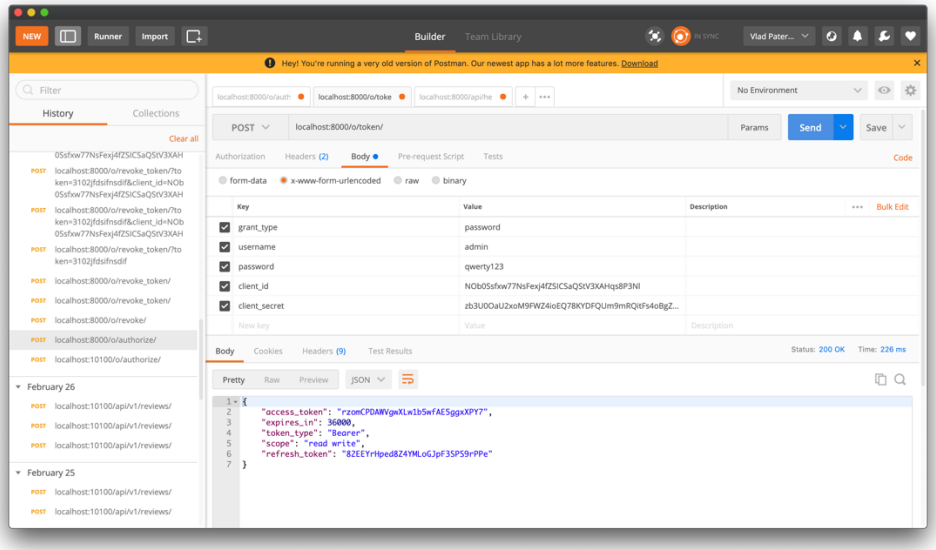


Рисунок 3.1 - Створення токена доступу

Щоб видалити токен, необхідно відправити токен на url /o/revoke_token/ з даними, вказаними на рисунку 3.2.

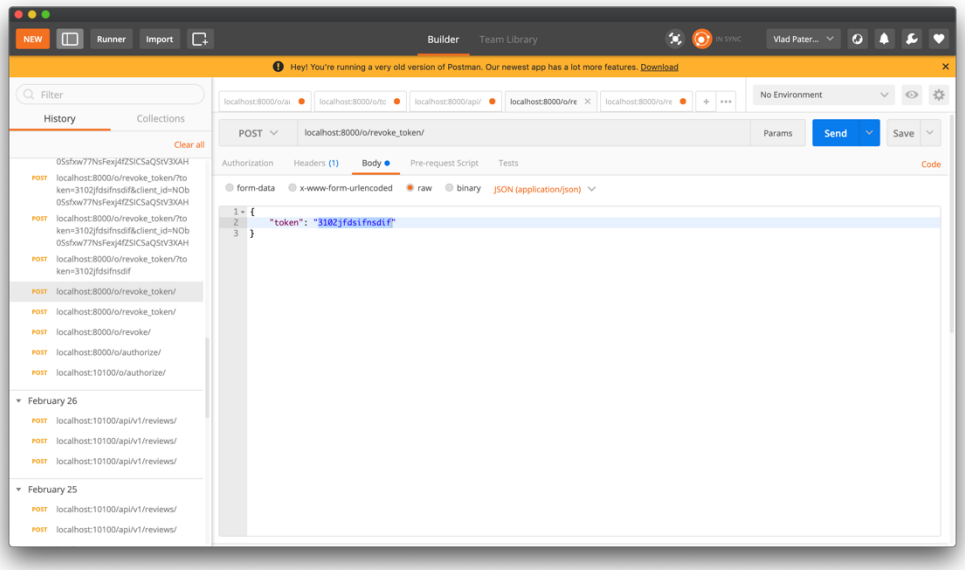


Рисунок 3.2 – Видалення токена доступу

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

СЕРВІС АВТОРИЗАЦІЇ ЗА ДОПОМОГОЮ СТАНДАРТУ OAUTH2

Графічний матеріал

КПІ.ІП-5116.045490.06.99

“ПОГОДЖЕНО”

Керівник проекту:

_____ О. Д. Фіногенов

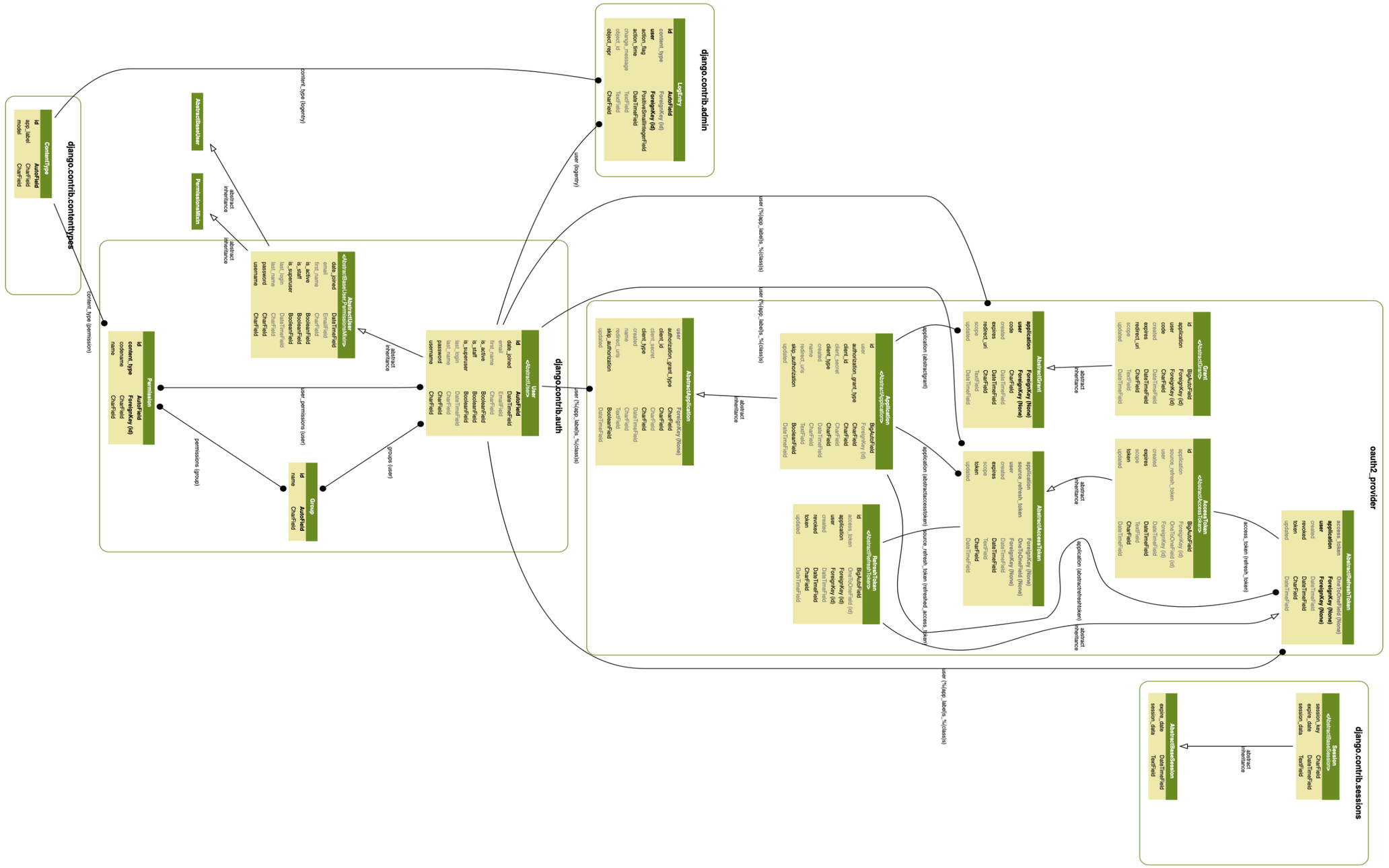
Нормоконтроль:

_____ М.М. Головченко

Виконавець:

_____В. В. Патерило

Київ – 2019 року



Змін.	Арк.	№ докумен	Підпис	Дата
Розробив		Гатерило В. В		
Перевірив		Фіногенов О.Д.		
Т. контр.				
Н. контр.		Голобченко М.М.		
Затвердив		Фіногенов О.Д.		

КПІ.ІП-5116.045490.06.99.СБД

Схема бази даних

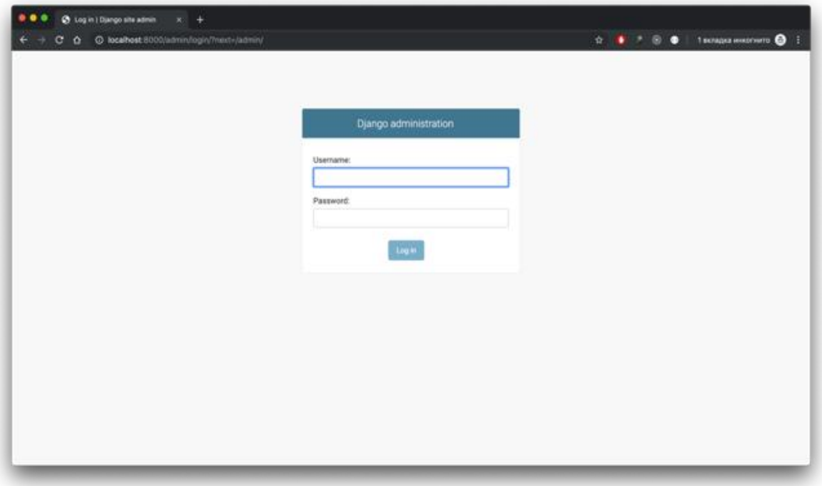
Літера	Маса	Масштаб

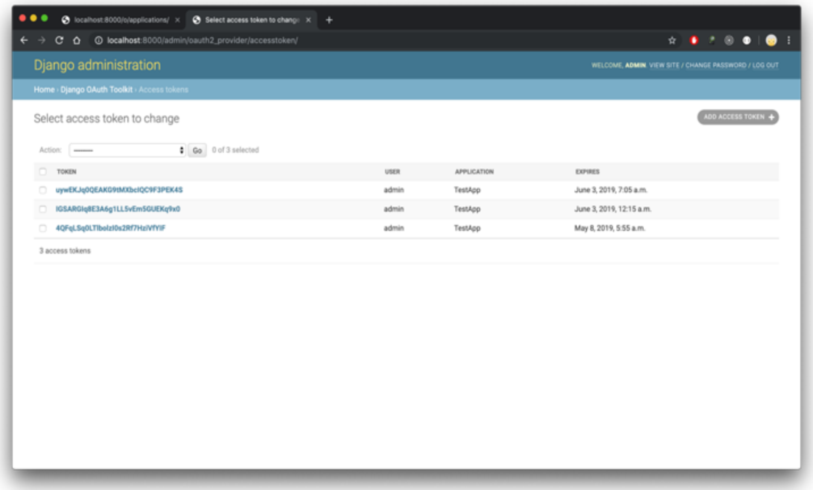
Арк/ш | Арк/шів

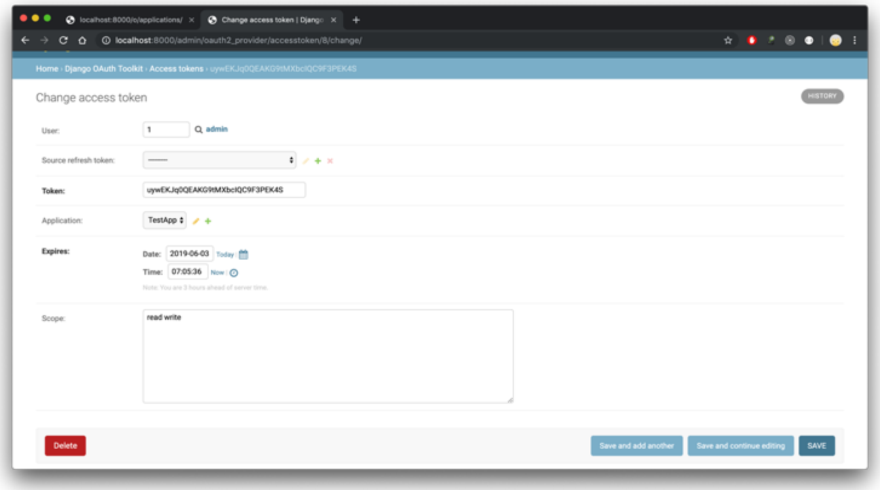
Сервіс авторизації за допомогою стандарту OAuth2

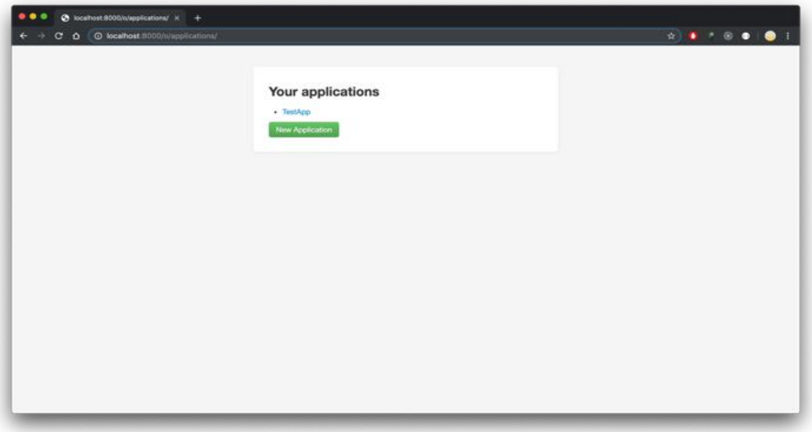
КПІ ім.Ігоря Сікорського
Кафедра АСОІУ
гр. ІП-51

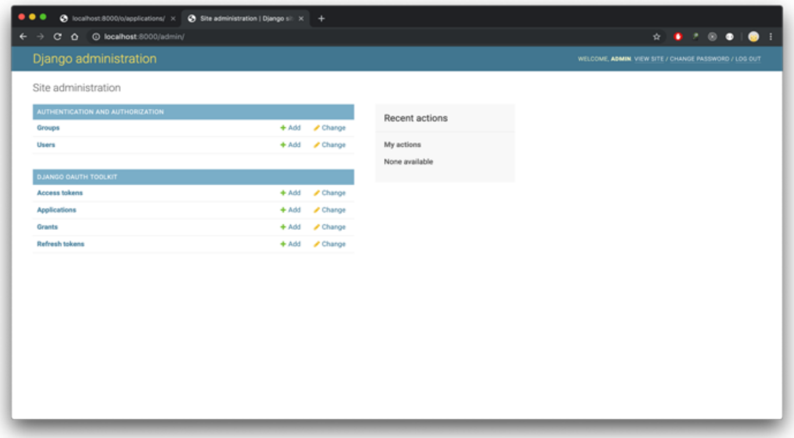
КПІ.ІП-5116.045490.06.99.КЕ

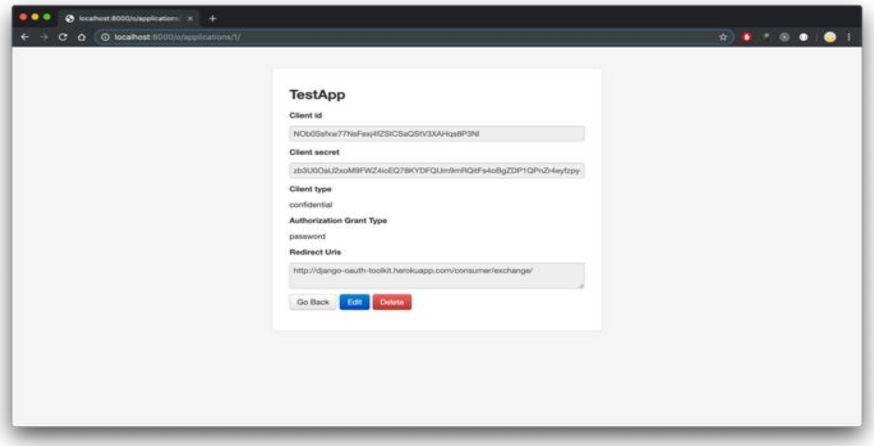


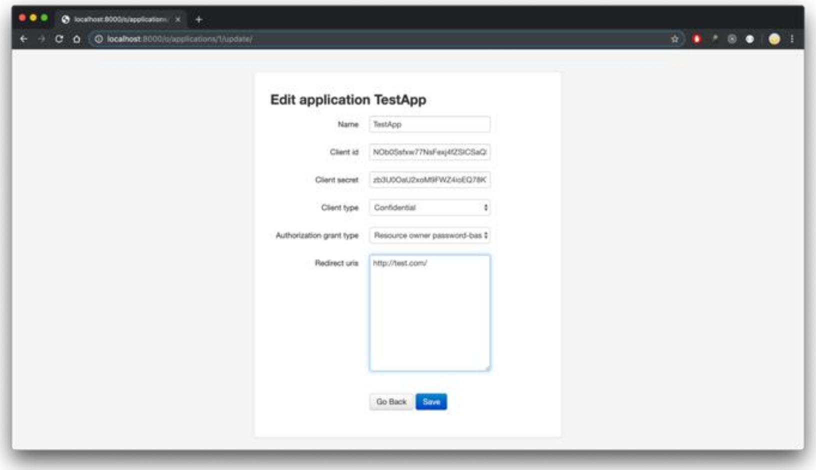


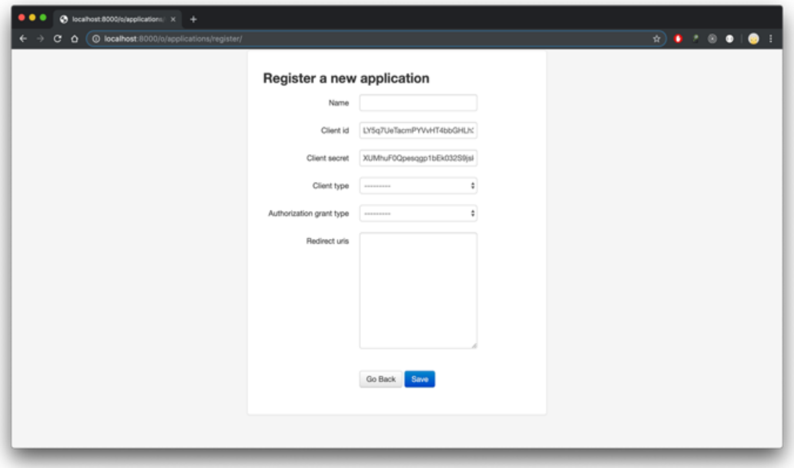


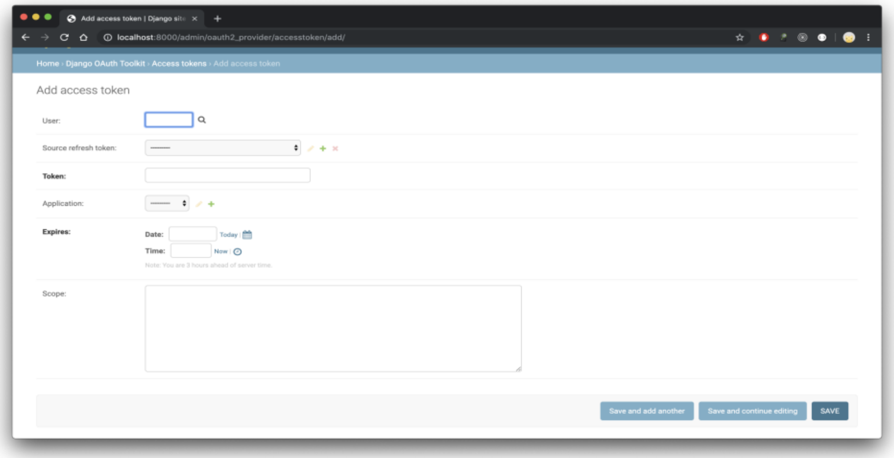




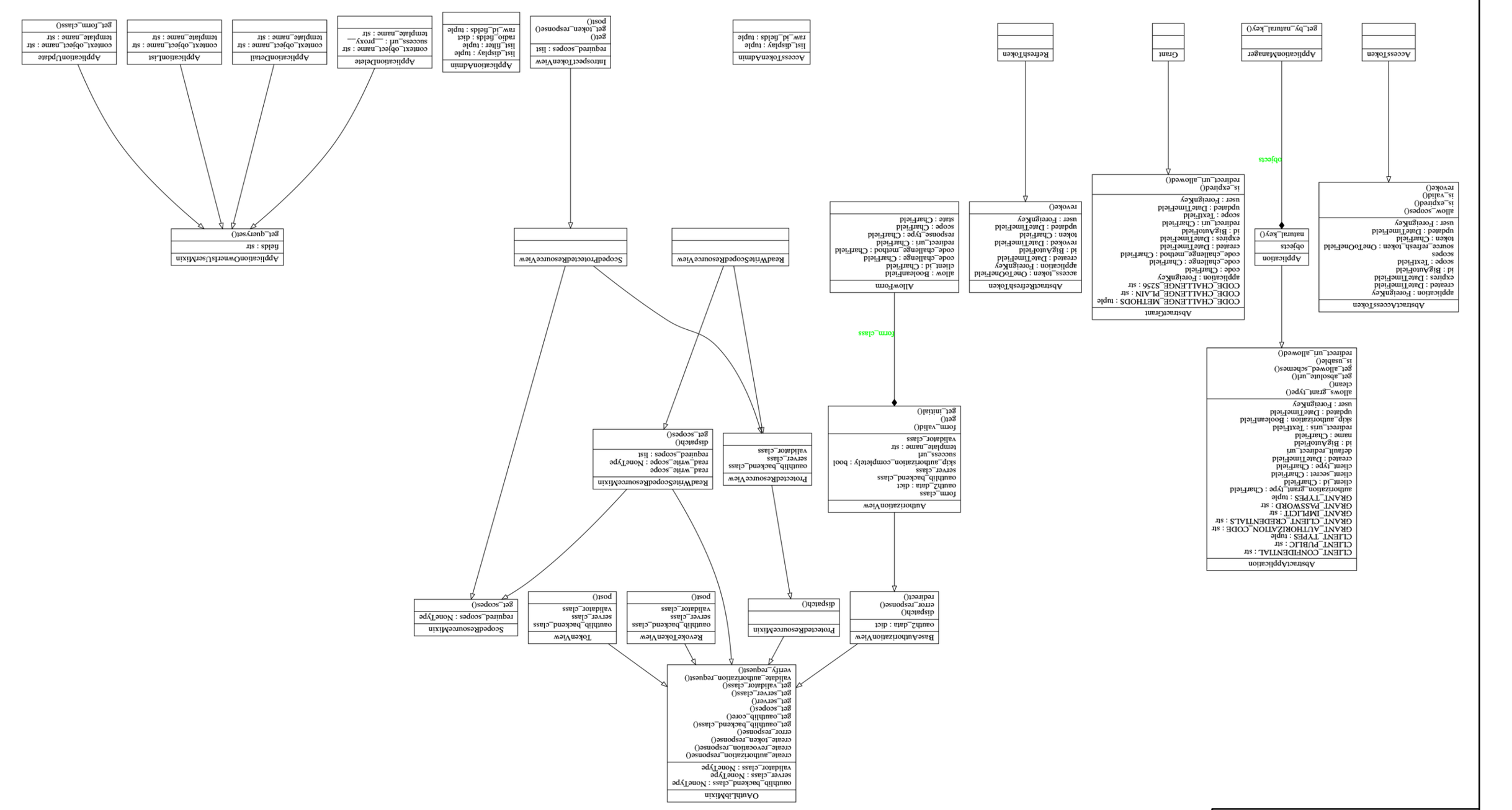








					КПІ.ІП-5116.045490.06.99.КЕ		
					Схема структурна екранних форм	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Патерило В. В.				Аркуш	Аркушів
Перевірив		Фіногенов О.Д.					
Т. кон.					Сервіс авторизації за допомогою стандарту OAuth2	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51	
Н. кон.		Головченко М.М.					
Затвердив		Фіногенов О.Д.					



3м.	Арк.	№ докумнта	Підпис	Дата	*КПІ.ІП-5116.045490.06.99.СС*
Розробив		Татарило В. В			
Перевірив		Фіногенов О.Д.			
Т. кон.					Схема структура класів програмного забезпечення
Н. кон.		Голованенко М.М.			
Затвердив		Фіногенов О.Д.			
					Сервіс авторизації за допомогою стандарту OAuth2
Літера	Маса	Масштаб			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
			Арк.	Аркушів	

КПІ.ІП-5116.045490.06.99.СС